IMPLEMENTASI ALGORITMA SHA-3 DAN MCELIECE DENGAN KODE HAMMING UNTUK OTENTIKASI DOKUMEN BERBASIS DIGITAL SIGNATURE

SKRIPSI

OLEH:
VADILLATUL NI'MA MAULIDYA
NIM. 210601110059

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

IMPLEMENTASI ALGORITMA SHA-3 DAN MCELIECE DENGAN KODE HAMMING UNTUK OTENTIKASI DOKUMEN BERBASIS DIGITAL SIGNATURE

SKRIPSI

Diajukan Kepada
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Matematika (S.Mat)

OLEH
VADILLATUL NI'MA MAULIDYA
NIM. 210601110059

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

IMPLEMENTASI ALGORITMA SHA-3 DAN MCELIECE DENGAN KODE HAMMING UNTUK OTENTIKASI DOKUMEN BERBASIS DIGITAL SIGNATURE

SKRIPSI

Oleh
Vadillatul Ni'ma Maulidya
NIM. 210601110059

Telah Disetujui Untuk Diuji

Malang, 14 Mei 2025

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Khudzaifah, M.Si.	Abdul Aziz, M.Si.
NIPPPK. 19900511 202321 1 029	NIP. 19760318 200604 1 002

Mengetahui,
Ketua Program Studi Matematika

Dr. Elly Susanti, M.Sc.
NIP. 19741129 200012 2 005

IMPLEMENTASI ALGORITMA SHA-3 DAN MCELIECE DENGAN KODE HAMMING UNTUK OTENTIKASI DOKUMEN BERBASIS DIGITAL SIGNATURE

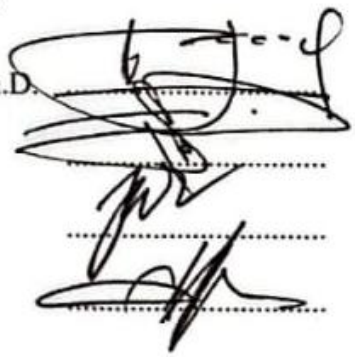
SKRIPSI

Oleh
Vadillatul Ni'ma Maulidya
NIM. 210601110059

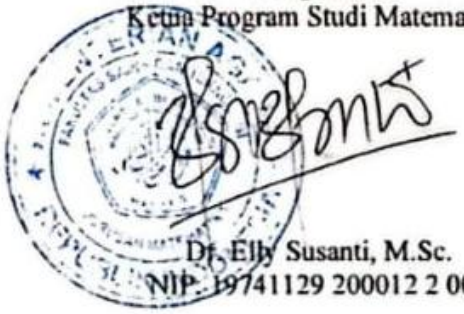
Telah Dipertahankan di Depan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Matematika (S.Mat)

Malang, 16 Juni 2025

Ketua Penguji	: Prof. Dr. H. Turmudi, M.Si., Ph.D.	.....
Anggota Penguji 1	: Hisyam Fahmi, M.Kom.	.....
Anggota Penguji 2	: Muhammad Khudzaifah, M.Si.	.....
Anggota Penguji 3	: Abdul Aziz, M.Si.	.....



Mengetahui,
Ketua Program Studi Matematika



Dr. Elly Susanti, M.Sc.
NIP. 19741129 200012 2 005

PERNYATAAN KEASLIAN TULISAN

Saya bertanda tangan di bawah ini

Nama : Vadillatul Ni'ma Maulidya

NIM : 210601110059

Program Studi : Matematika

Fakultas : Sains dan Teknologi

Judul Skripsi : Implementasi Algoritma SHA-3 dan McEliece dengan Kode Hamming untuk Otentikasi Dokumen Berbasis *Digital Signature*

Menyatakan bahwa skripsi yang saya tulis ini merupakan hasil karya sendiri, bukan mengambil alih tulisan atau pemikiran orang lain. Saya menyatakan skripsi ini sebagai pemikiran saya, kecuali dengan mencantumkan sumber kutipan pada daftar rujukan di halaman terakhir. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil plagiasi atau tiruan orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 16 Juni 2025

Vadillatul Ni'ma Maulidya

NIM. 210601110059

MOTO

ذنب العالم ذنب واحد وذنب الجاهل ذنبان

“dosa orang berilmu itu satu, dosa orang bodoh itu dua”

– رواه الديلمى –

“sukses ada karena usaha saya, bukan hanya karena takdir”

- Vadillatul -

PERSEMBAHAN

بسم الله الرحمن الرحيم

Skripsi ini dipersembahkan kepada:
Ibunda Jumiati, yang selalu mencurahkan segala bentuk cinta kasihnya tanpa meminta balasan, Almarhum Ayahanda Ahmad Naim yang telah memberi semangat dan membuka ilmu pengetahuan sampai pada tahap ini, serta kedua saudara yang selalu memberikan doa dan dukungan terbaik kepada penulis.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Segala puji bagi Allah *Subhanahu Wa Ta'ala* yang telah melimpahkan rahmat, taufik, dan hidayah-Nya, sehingga penulis senantiasa diberikan kesehatan dan kesempatan untuk menyelesaikan proposal dengan judul "Implementasi Algoritma SHA-3 dan McEliece dengan Kode Hamming untuk Otentikasi Dokumen Berbasis *Digital Signature*" sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang matematika di Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Sholawat dan salam senantiasa tercurahkan kepada junjungan kita Nabi Muhammad *Shallallahu 'Alaihi Wasallam*, yang telah membawa kita dari zaman jahiliah menuju zaman yang penuh rahmat yakni zaman Islamiah.

Penulis mengucapkan terima kasih kepada berbagai pihak yang telah membimbing dan membantu dalam penyusunan proposal ini, sehingga proposal ini dapat diselesaikan sesuai yang diharapkan. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Prof. Dr. H. M. Zainuddin, MA. selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prof. Dr. Sri Harini, M.Si. selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Elly Susanti, M.Sc. selaku Ketua Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Muhammad Khudzaifah, M.Si. selaku Dosen Pembimbing I yang telah memberikan banyak ilmu, arahan, masukan, dan nasihat kepada penulis.
5. Abdul Aziz, M.Si. selaku Dosen Pembimbing II yang telah memberikan banyak ilmu sekaligus telah sabar memberikan arahan, masukan, dan nasihat.
6. Seluruh sivitas akademika Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, terutama jajaran dosen yang telah memberikan pengalaman perkuliahan yang luar biasa.

7. Ibunda, Jumiati, yang selalu memberikan doa terbaik dan waktu berharga untuk mendengarkan semua keluh kesah.
8. Almarhum ayahanda, Ahmad Naim, yang telah memberikan kenangan dan amanah untuk menyelesaikan pendidikan ini.
9. Kedua saudara yang telah memberikan dukungan dan doa.
10. Seluruh teman-teman, baik di Program Studi Matematika maupun di luar Program Studi yang tidak dapat peneliti sebutkan satu persatu, yang telah menemani menyelesaikan penelitian.
11. Kepada idola dan karakter anime yang telah memberi semangat dan kebahagiaan selama menyelesaikan penelitian.
12. Semua pihak yang secara langsung atau tidak langsung telah ikut memberikan bantuan dalam menyelesaikan penelitian.

Semoga Allah *Subhanahu Wa Ta'ala* melimpahkan rahmat dan karunia-Nya kepada kita semua. Penulis berharap skripsi ini dapat bermanfaat tidak hanya bagi penulis, tetapi juga bermanfaat bagi semua pihak yang membacanya. *Aamin Allahumma Aamiin*.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Malang, 16 Juni 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR SIMBOL

n : Parameter panjang kolom matriks

k : Parameter dimensi baris matriks

t : Jumlah kesalahan yang dapat dikoreksi

S : Matriks pengacak

G : Matriks generator

M : Matriks permutasi

I : Matriks identitas

H : Matriks *parity-chec*

P : Sub-matriks paritas

G' : Kunci privat dari perkalian $M \cdot G \cdot P$

H' : Kunci privat dari perkalian $H^T \cdot M$

c : *ciphertext*

c' : Dekripsi *ciphertext*

m : Pesan asli / vektor pesan yang akan dikirim

m' : Pesan asli / vektor pesan yang telah di dekripsi

e : *error* untuk *encoding*

h : *noise* untuk *digital signature*

ABSTRAK

Maulidya, Vadillatul Ni'ma. 2025. **Implementasi Algoritma SHA-3 dan McEliece dengan Kode Hamming untuk Otentikasi Dokumen Berbasis *Digital Signature***. Skripsi. Jurusan Matematika fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Muhammad Khudzaifah, M.Si. (II) Abdul Aziz, M.Si.

Kata Kunci: algoritma SHA-3, algoritma McEliece, dokumen digital, *message digest*, *digital signature*

Perlindungan data pribadi merupakan langkah penting untuk menjaga keamanan dan kerahasiaan informasi, termasuk dokumen dalam bentuk digital. Salah satu metode yang dapat digunakan untuk menjaga keautentikan dokumen digital yaitu dengan *digital signature*. *Digital signature* dibuat menggunakan fungsi *hash* SHA-3 dan kombinasi algoritma kriptografi McEliece dengan kode Hamming. Proses diawali dengan melakukan *hashing* terhadap isi dokumen menggunakan SHA-3 untuk menghasilkan *message digest* yang telah dikonversi dalam bentuk biner dan dienkripsi menggunakan algoritma McEliece untuk menghasilkan *digital signature*. Dokumen dapat dianggap valid ketika diverifikasi apabila menghasilkan *digital signature* yang sama dengan nilai *hash* dokumen asli. Penelitian menunjukkan bahwa *digital signature* yang dihasilkan akan bergantung pada kunci McEliece yang digunakan, sehingga dapat berbeda untuk setiap dokumen. Kombinasi antara algoritma SHA-3 dan McEliece menunjukkan Efek Avalanche yang mendekati 50%, hal ini menunjukkan bahwa sistem memiliki sifat difusi yang baik. Dengan demikian, kombinasi algoritma SHA-3 dan McEliece dapat diandalkan sebagai solusi kriptografi dalam menjamin keaslian dan perlindungan dokumen digital.

ABSTRACT

Maulidya, Vadillatul Ni'ma. 2025. **Implementation of SHA-3 and McEliece Algorithms with Hamming Code for *Digital Signature* Based Document Authentication**. Thesis. Department of Mathematics, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Advisors: (I) Muhammad Khudzaifah, M.Si. (II) Abdul Aziz, M.Si.

Keywords: SHA-3 algorithm, McEliece algorithm, digital document, message digest, digital signature

Personal data protection is an important step to maintain the security and confidentiality of information, including documents in digital form. One method that can be used to maintain the authenticity of digital documents is with a digital signature. Digital signature is created using SHA-3 hash function and combination of McEliece cryptography algorithm with Hamming code. The process begins by hashing the contents of the document using SHA-3 to produce a message digest that has been converted in binary form and encrypted using the McEliece algorithm to produce a digital signature. The document can be considered valid when verified if it produces the same digital signature as the hash value of the original document. Research shows that the resulting digital signature will depend on the McEliece key used, so it can be different for each document. Combination of the SHA-3 and McEliece algorithms demonstrates an Avalanche Effect approaching 50%, indicating that the system has good diffusion properties. Thus, the combination of SHA-3 and McEliece algorithms can be relied upon as a cryptographic solution in guaranteeing the authenticity and protection of digital documents.

مستخلص البحث

موليدية، فضيلة النعمة. ٢٠٢٥. **تنفيذ خوارزمية SHA-3 وخوارزمية McEliece مع كود هامينج للتوثيق المستند الرقمي إلى التوقيع الرقمي**. البحث الجامعي. قسم الرياضيات، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرفة: (١) محمد حذيفة، الماجستر فى العلوم. (٢) عبد العزيز، الماجستر فى العلوم.

الكلمات المفتاحية: خوارزميات SHA-3، وخوارزميات McEliece، المستند الرقمي، ملخص الرسالة، التوقيع الرقمي

تعد حماية البيانات الشخصية خطوة مهمة للحفاظ على أمن وسرية المعلومات، بما في ذلك المستندات في شكل رقمي. إحدى الطرق التي يمكن استخدامها للحفاظ على صحة المستندات الرقمية هي التوقيع الرقمي. يتم إنشاء التوقيع الرقمي باستخدام دالة تجزئة *SHA-3* ومزيج من خوارزمية التشفير *McEliece* مع كود هامينج. تبدأ العملية عن طريق تجزئة محتويات المستند باستخدام *SHA-3* لإنتاج ملخص رسالة تم تحويله في شكل ثنائي وتشفيره باستخدام خوارزمية McEliece لإنتاج توقيع رقمي. يمكن اعتبار المستند صالحًا عند التحقق من صحته إذا أنتج توقيعًا رقميًا يماثل قيمة تجزئة المستند الأصلي. تُظهر الأبحاث أن التوقيع الرقمي الذي تم إنشاؤه يعتمد على مفتاح *McEliece* المستخدم، لذلك يمكن أن يكون مختلفًا لكل مستند. ، يُظهر الجمع بين خوارزميات *SHA-3* وخوارزميات *McEliece* تأثير الانهيار الجليدي الذي يقترب من ٥٠ %، مما يشير إلى أن النظام يتمتع بخصائص انتشار جيدة. وبالتالي، يمكن الاعتماد على الجمع بين خوارزمية *SHA-3* وخوارزمية *McEliece* كحل تشفيري لضمان صحة وحماية المستندات الرقمية.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Berdasarkan Undang-Undang Nomor 27 Tahun 2022 tentang Perlindungan Data Pribadi (UU PDP), perlindungan data pribadi merupakan langkah untuk menjaga keamanan dan kerahasiaan informasi individu selama proses pengumpulan, penyimpanan, pemrosesan, dan penyebaran data. Undang-undang ini dirancang untuk melindungi hak-hak konstitusional pemilik data pribadi, mencegah penyalahgunaan, dan memastikan bahwa data pribadi hanya digunakan untuk tujuan yang sah dan sesuai izin yang berwenang. Data tersebut mencakup informasi dalam dokumen keuangan, kesehatan, perbankan, hukum, dan lain sebagainya. Informasi dalam dokumen memerlukan perlindungan yang cukup karena data di dalamnya dapat dengan mudah disebarluaskan, disalin atau diubah oleh pihak yang tidak bertanggung jawab.

Penerapan sistem otentikasi dokumen yang kuat menjadi salah satu cara untuk mengamankan dokumen. Otentikasi pada dokumen penting dilakukan untuk memastikan integritas, keaslian, dan keamanan data yang disimpan atau ditransmisikan melalui saluran komunikasi seperti internet yang kerap disebut dokumen digital. Berbagai metode otentikasi telah dikembangkan untuk mengatasi permasalahan ini, di antaranya otentikasi berbasis kode QR (*Quick Response Code*), *watermarking*, dan *digital signature*. Salah satu metode yang banyak digunakan adalah *digital signature* yang memanfaatkan prinsip kriptografi untuk memberikan bukti autentikasi yang kuat terhadap dokumen elektronik.

Menjaga integritas dan keaslian informasi juga di ajarkan Islam dalam bentuk amanah. Hal ini sebagaimana dijelaskan dalam firman Allah SWT di dalam Al-Qur'an yang artinya (Kementerian Agama, 2022a):

"(Sungguh beruntung pula) orang-orang yang memelihara amanat dan janji mereka." (QS. Al-Mu'minun: 8)

Ayat ini mengajarkan setiap umat muslim untuk bertanggung jawab menjaga privasi dan keamanan data yang dipercayakan, sehingga amanah tersebut tidak disalahgunakan oleh pihak lain. Selain itu, dalam QS. An-Nur ayat 27, Allah SWT memerintahkan agar tidak melanggar hak-hak privasi orang lain tanpa izin, menegaskan pentingnya menghormati batasan privasi dan hak individu dalam kehidupan sehari-hari (Kementerian Agama, 2022b).

"Wahai orang-orang yang beriman, janganlah memasuki rumah yang bukan rumahmu sebelum meminta izin dan memberi salam kepada penghuninya. Demikian itu lebih baik bagimu agar kamu mengambil pelajaran." (QS. An-Nur: 27).

Ayat ini menekankan adab sosial dan etika seorang mukmin dalam memasuki rumah orang lain. Allah memerintahkan untuk meminta izin dan mengucapkan salam terlebih dahulu sebelum masuk. Hal ini bertujuan untuk menjaga privasi dan kehormatan penghuni rumah (Az-Zuhaili, 2016). Prinsip ini mendasari penerapan sistem yang dapat melindungi informasi dengan aman dan memastikan bahwa dokumen yang beredar dapat dipercaya. Seiring dengan perkembangan teknologi penerapan *digital signature* menjadi langkah yang tepat untuk memastikan informasi yang disampaikan tetap terjaga, sekaligus mencerminkan nilai-nilai keislaman tentang kejujuran dan tanggung jawab dalam menjaga amanah.

Digital signature merupakan sebuah mekanisme kriptografi yang dapat memverifikasi keaslian integritas, dan otentikasi suatu dokumen digital. Berbeda

dengan tanda tangan konvensional yang berbentuk tulisan tangan, *digital signature* menggunakan algoritma matematika untuk menghasilkan nilai unik yang terkait langsung dengan isi dokumen. Nilai ini bersifat unik dan akan berubah jika terdapat modifikasi pada dokumen, sehingga memungkinkan pendeteksian perubahan yang tidak sah. Seiring dengan perkembangan teknologi keamanan siber, berbagai algoritma telah digunakan dalam pembuatan *digital signature,* seperti algoritma SHA dan MD5. Namun, dalam segi efisiensi dan keamanan, algoritma SHA memiliki ketahanan yang lebih baik dibandingkan dengan MD5. MD5 diketahui rentan terhadap serangan, di mana dua dokumen yang berbeda dapat memiliki nilai *hash* yang sama, sehingga membuatnya kurang dapat diandalkan untuk tujuan keamanan tingkat tinggi (Santoso dkk., 2019).

Algoritma kriptografi yang paling populer digunakan untuk menjaga keamanan dan integritas data digital berbasis *digital signature* saat ini yaitu, *Secure Hash Algorithm* (SHA). *Hash* adalah fungsi satu arah yang dirancang untuk menerima masukan berupa *string* dengan panjang sembarang dan mengubahnya menjadi keluaran (*massage digest*) dengan panjang tetap (*fixed*). Selain itu, SHA memiliki berbagai jenis algoritma diantaranya SHA-0, SHA-1, SHA-2, dan SHA-3, di mana masing-masing algoritma tersebut memiliki tingkatan keamanan yang berbeda (Sari, 2021).

Algoritma SHA-3 dipilih dalam penelitian ini karena menawarkan ketahanan dan keamanan data yang lebih tinggi dibandingkan algoritma sebelumnya, seperti SHA-0, SHA-1 dan SHA-2. Selain itu, dengan fleksibilitas ukuran *output*, seperti 224-bit, 256-bit, 384-bit, dan 512-bit, SHA-3 menjadi pilihan utama dalam sistem keamanan tingkat tinggi (Kurniawan, Kusyanti, & Nurwarsito, 2017). Oleh

karenanya, SHA-3 dengan *output* 256-bit dipilih pada penelitian ini karena menawarkan keseimbangan optimal antara keamanan dan efisiensi. Selain itu, panjang *hash* 256-bit sudah cukup kuat untuk mencegah serangan, sekaligus lebih efisien dalam hal penggunaan ruang penyimpanan dan kecepatan pemrosesan dibandingkan varian dengan *output* lebih panjang seperti 384-bit atau 512-bit. Penggunaan *output* 256-bit juga didasarkan pada kompatibilitasnya dengan berbagai algoritma lain, seperti SHA-256 dan BLAKE2, yang umum digunakan dalam berbagai aplikasi keamanan dan kriptografi.

Selain memanfaatkan SHA-3 untuk menghasilkan kombinasi fungsi *hash*, metode kriptografi tambahan seperti skema tanda tangan McEliece berperan penting dalam keamanan digital. Algoritma McEliece merupakan salah satu algoritma kriptografi kunci publik berbasis kode *error correction* yang menggunakan prinsip matriks biner untuk mengenkripsi pesan. Meskipun ukuran kunci algoritma McEliece cenderung lebih besar dibandingkan algoritma lain seperti RSA (Rivest-Shamir-Adleman) dan ECC (*Elliptic Curve Cryptography*). Hal ini justru menjadi salah satu keunggulannya dalam menghadapi ancaman komputasi kuantum. Sehingga, algoritma ini menjadi kandidat yang kuat dalam pengembangan sistem keamanan digital di masa depan terutama dalam perlindungan dokumen dan transaksi elektronik (Ilmiyah, 2018).

Salah satu kode yang dapat digunakan dalam skema McEliece adalah kode Hamming yang memanfaatkan konsep paritas dan bit paritas untuk memvalidasi suatu data yang lebih sederhana. Kode ini dirancang untuk mendeteksi dan memperbaiki kesalahan bit, sehingga sering digunakan dalam sistem komunikasi dan penyimpanan data. Oleh karena itu, hubungan ini digunakan dalam penelitian

untuk menunjukkan bagaimana teknik *error correction* kode Hamming dapat diintegrasikan ke dalam skema enkripsi untuk meningkatkan ketahanan dan keandalan dalam komunikasi digital. Skema ini diharapkan mampu meningkatkan ketahanan komunikasi digital terhadap serangan serta menjaga keakuratan dan keamanan data yang ditransmisikan.

Dalam penelitian ini, kombinasi algoritma *digital signature* berbasis SHA-3 dan McEliece diharapkan mampu menciptakan sistem otentikasi yang kuat dalam aspek keamanan maupun integritas dokumen digital. Penelitian ini bertujuan untuk mengembangkan sistem otentikasi dokumen berbasis *digital signature* yang memanfaatkan algoritma SHA-3 untuk menghasilkan fungsi *hash* yang unik, serta McEliece sebagai algoritma untuk mengamankan fungsi *hash*. Selain itu, penerapan kode Hamming dalam skema McEliece berfungsi untuk mendeteksi dan mengoreksi kesalahan selama proses pengiriman atau penyimpanan. Dengan demikian, penelitian ini difokuskan pada penerapan SHA-3 dan McEliece dengan kode Hamming dalam otentikasi dokumen digital berbasis *digital signature*, guna meningkatkan keamanan dan keandalan sistem otentikasi di era digital.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini, sebagai berikut:

1. Bagaimana proses otentikasi dokumen dan pengaruh *digital signature* pada perubahan kunci menggunakan algoritma SHA-3 dan kriptografi McEliece dengan kode Hamming pada suatu dokumen digital?

2. Bagaimana hasil verifikasi *digital signature* dalam mendeteksi perubahan isi dokumen menggunakan algoritma SHA-3 dan kriptografi McEliece dengan kode Hamming?
3. Bagaimana hasil evaluasi algoritma SHA-3 dan kriptografi McEliece dengan kode Hamming dalam meningkatkan keamanan dan efisiensi otentikasi dokumen digital?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan, penelitian ini memiliki beberapa tujuan yang ingin dicapai, sebagai berikut:

1. Untuk menganalisis proses otentikasi dokumen dan pengaruh *digital signature* pada perubahan kunci menggunakan algoritma SHA-3 dan McEliece dengan kode Hamming pada suatu dokumen digital.
2. Untuk menganalisis hasil verifikasi *digital signature* dalam mendeteksi perubahan isi dokumen menggunakan algoritma SHA-3 dan kriptografi McEliece dengan kode Hamming.
3. Untuk mengevaluasi algoritma SHA-3 dan kriptografi McEliece dengan kode Hamming dalam meningkatkan keamanan dan efisiensi otentikasi dokumen digital.

1.4 Manfaat Penelitian

Berdasarkan tujuan penelitian yang telah diuraikan, manfaat penelitian ini dapat dibedakan berdasarkan kepentingan berbagai pihak, diantaranya:

1. Manfaat bagi peneliti

   Penelitian ini merupakan salah satu upaya mengembangkan pengetahuan mengenai algoritma SHA-3 dan kriptografi McEliece dengan kode Hamming.

2. Manfaat bagi pembaca

   Penelitian ini dapat dijadikan sebagai referensi untuk penelitian selanjutnya yang berkaitan dengan *digital signature*.

3. Manfaat bagi instansi

   Penelitian ini dapat dijadikan bahan pengembangan akreditasi atau kurikulum keilmuan matematika terutama dalam bidang aljabar.

1.5 Batasan Masalah

Agar penelitian ini tidak melampaui cakupan yang diperlukan, maka diperlukan adanya batasan masalah, sebagai berikut:

1. Menghitung keluaran dari fungsi *hash* (*message digest*) menggunakan algoritma SHA-3 dengan dokumen yang digunakan sebanyak dua puluh dokumen berformat .pdf, berisi alfabet, angka, dan karakter.
2. Penelitian ini berfokus pada penggunaan algoritma SHA-3 dengan *output* sepanjang 256-bit.

1.6 Definisi Istilah

Istilah yang akan digunakan pada penulisan ini, diantaranya:

Digital signature : Tanda tangan digital yang mampu memverifikasi keaslian suatu dokumen digital.

Dokumen digital	:	Dokumen dalam bentuk elektronik yang disimpan, dikirim, dan diproses melalui media digital.
Dokumen berbasis *digital signature*	:	Dokumen digital yang dilengkapi dengan tanda tangan elektronik untuk menjamin keaslian, integritas, dan legalitasnya melalui mekanisme kriptografi.
Otentikasi	:	Proses untuk membuktikan bahwa suatu dokumen berasal dari sumber yang sah.
Hash function	:	Fungsi satu arah yang dirancang untuk menerima masukan berupa *string* dengan panjang sembarang dan mengubahnya menjadi keluaran (*message digest*) dengan panjang tetap (*fixed*).
Message digest	:	Hasil keluaran dari proses *hashing*, berupa kombinasi heksadesimal.
Signing	:	Proses penandatanganan dokumen digital sebelum dikirim.
Verifying	:	Proses memverifikasi dokumen digital yang telah diterima.
Enkripsi atau *encoding*	:	Proses mengacak atau mengubah pesan asli sehingga menghasilkan pesan yang sulit dibaca.
Dekripsi atau *decoding*	:	Proses memulihkan pesan teracak sehingga menghasilkan pesan asli seperti semula.
Plaintext	:	Pesan asli yang dapat dibaca dengan mudah.
Ciphertext	:	Pesan teracak yang telah diubah dan sukar dibaca.

BAB II

KAJIAN TEORI

2.1 Algoritma Kriptografi

Kriptografi berasal dari bahasa Yunani, *cryptos* berarti tersembunyi atau rahasia dan *graphein* berarti menulis atau tulisan. Bruce Schneier, dalam bukunya *Applied Cryptography* (1996), mendefinisikan kriptografi sebagai seni dan ilmu untuk menjaga keamanan pesan (*Cryptography is the art and science of keeping messages secure*) (Nurhayati, Kastari, & Fahrianto, 2022). Dengan demikian, kriptografi merupakan ilmu dan praktik untuk mengamankan informasi melalui teknik pengkodean sehingga hanya pihak yang berwenang yang dapat mengakses pesan tersebut. Selain itu, kriptografi juga mempelajari teknik-teknik yang harus diperhatikan dalam mengamankan pesan, yaitu (Munir, 2019):

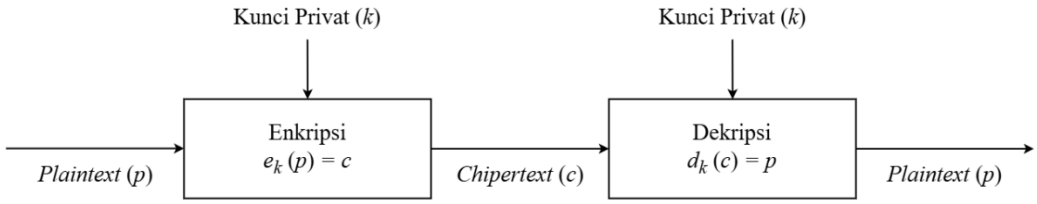
1. *Confidentiality* (kerahasiaan): layanan ini berfungsi untuk melindungi pesan agar hanya dapat diakses oleh pihak yang berwenang.
2. *Data integrity* (integritas data): layanan ini memastikan pesan yang diterima merupakan pesan asli. Oleh sebab itu, sistem harus dapat mendeteksi perubahan, seperti mengganti, menghapus, menyisipkan, atau modifikasi lain pada pesan yang diterima.
3. *Authentication* (otentikasi): layanan ini berperan dalam mengidentifikasi keaslian pesan yang dikirim atau diterima dari pihak-pihak yang berkomunikasi. Dengan otentikasi, sistem dapat menjamin bahwa pesan benar-benar berasal dari pengirim yang sah dan diterima oleh penerima yang dituju.

4. *Non-repudiation* (anti penyangkalan): layanan ini dirancang untuk mencegah penolakan dari pihak yang terlibat dalam komunikasi. Misalnya, pengirim menyangkal telah mengirim pesan atau penerima menyatakan tidak menerima pesan.

Prinsip dasar kriptografi merupakan proses enkripsi dan dekripsi untuk mengubah pesan menjadi bentuk pesan yang abstrak dan tidak mudah dipahami. Enkripsi adalah proses mengacak pesan asli atau *plaintext*, sehingga menghasilkan pesan acak yang sulit dibaca. Sedangkan, dekripsi adalah proses memulihkan pesan teracak atau *ciphertext*, sehingga menghasilkan pesan asli seperti semula. Proses ini dapat dilakukan pada pesan yang sedang dikirim maupun pesan yang disimpan.

2.1.1 Algoritma Kriptografi Simetris

Kriptografi simetris dikenal juga dengan kriptografi konvensional yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Artinya, kunci yang digunakan untuk mengacak pesan (enkripsi) sama dengan kunci yang digunakan untuk memulihkan pesan (dekripsi). Metode ini dikenal sebagai kunci privat, karena menggunakan kunci yang sama untuk kedua proses. Secara sederhananya, proses tersebut digambarkan sebagai berikut:

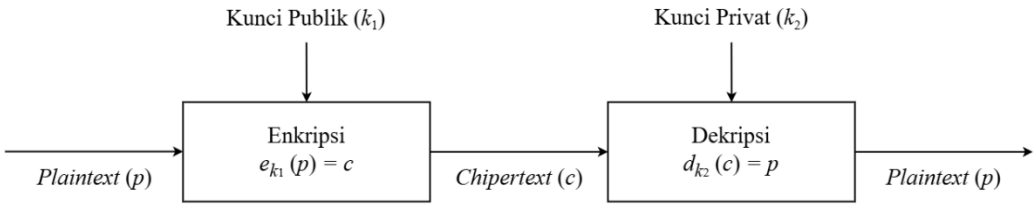


Gambar 2.1 Algoritma Kriptografi Simetris

Penerapan algoritma kriptografi simetris diantaranya, AES (*Advanced Encryption Standard*), DES (*Data Encryption Standard*), MARS, RC5 (*Rivest Code* 5), dan RC6 (*Rivest Code* 6). Dalam perkembangannya, kunci simetris memiliki beberapa keunggulan, salah satunya yaitu kecepatan operasi yang lebih tinggi. Namun, hal ini berkorelasi dengan penambahan ukuran pesan yang dikirim, karena kecepatan proses enkripsi dan dekripsi bergantung pada ukuran pesan. Oleh sebab itu, setiap proses enkripsi dan dekripsi memerlukan waktu yang lebih lama untuk pesan yang lebih besar. Selain itu, setiap pengguna yang berbeda memerlukan kunci yang berbeda untuk mengirim pesan. Hal ini, menyebabkan masalah dalam pengiriman dan manajemen kunci, yang biasa disebut dengan "*key distribution problem*" (Basri, 2016).

2.1.2 Algoritma Kriptografi Asimetris

Kriptografi asimetris dikenal juga dengan kriptografi kunci publik yang menggunakan sepasang kunci, yaitu kunci publik dan kunci privat. Kunci publik dapat dibagikan kepada siapa saja, sementara kunci privat harus dijaga kerahasiaannya. Prinsip dasar algoritma ini adalah kunci privat hanya dapat mendekripsi pesan yang dienkripsi dengan kunci publik dan sebaliknya. Hal ini memungkinkan pertukaran pesan yang aman tanpa harus mengirimkan kunci privat secara terpisah. Oleh sebab itu, teknik enkripsi kunci asimetris memiliki kesan yang lebih lambat dibanding menggunakan kunci simetris. Secara sederhananya, proses tersebut digambarkan sebagaimana ditunjukkan pada Gambar 2.2.



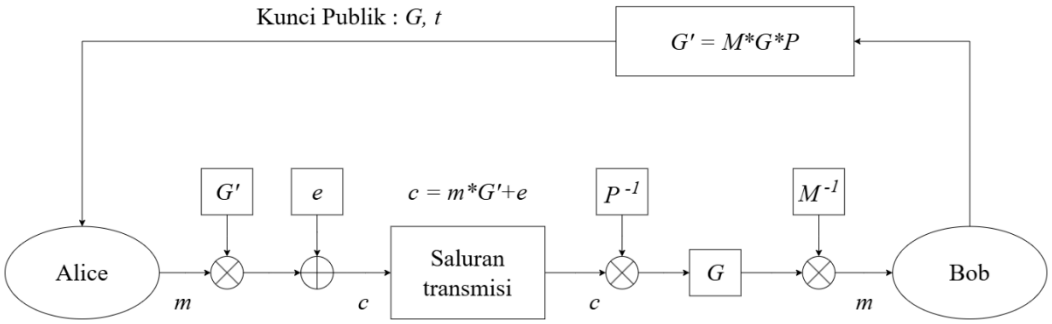
Gambar 2.2 Algoritma Kriptografi Asimetris

Penerapan algoritma kriptografi asimetris yang paling terkenal adalah RSA (Rivest-Shamir-Adleman), yang kembangkan pada tahun 1978 oleh Ron Rivest, Adi Shamir, dan Leonard Adleman. Namun, komputer kuantum memiliki potensi untuk meruntuhkan keamanan algoritma RSA. Selain itu, algoritma yang rentan terhadap komputer kuantum diantaranya, ECC (*Elliptic Curve Cryptography*), ElGamal dan Diffie-Hellman. Di sisi lain, masih banyak algoritma yang aman terhadap serangan komputer kuantum, antara lain BIKE (*Bit Flipping Key Encapsulation*), HQC (*High-Quality-Check*), dan McEliece (Kumar, 2022).

2.2 Algoritma Kriptografi McEliece

Algoritma McEliece merupakan salah satu skema kriptografi asimetris yang dirancang untuk menyediakan keamanan terhadap serangan komputasi kuantum. Robert J. McEliece pada tahun 1978 memperkenalkan algoritma McEliece sebagai algoritma berbasis kode *error correcting*. Salah satu kode yang dapat digunakan dalam skema McEliece adalah kode Hamming, yang merupakan jenis kode koreksi kesalahan linear yang mampu mendeteksi dan memperbaiki kesalahan selama proses transmisi data. Kode Hamming memiliki struktur yang memungkinkan koreksi satu bit kesalahan dalam setiap blok data, sehingga meningkatkan keandalan komunikasi dan keamanan dalam enkripsi pesan (Jaameri, 2019).

Algoritma McEliece sangat bergantung pada operasi matriks dalam proses enkripsi dan dekripsi. Oleh karena itu, skema ini umumnya menggunakan tiga parameter utama, yaitu (n, k, t). Parameter n menunjukkan panjang kode atau jumlah bit dalam satu blok *ciphertext*, sedangkan k menunjukkan dimensi kode sekaligus jumlah bit dalam pesan yang akan dienkripsi. Sementara, t menunjukkan jumlah bit kesalahan maksimum yang dapat dikoreksi oleh kode linier. Secara sederhana, algoritma McEliece dapat digambarkan sebagaimana ditunjukkan pada Gambar 2.3 (Baldi, Bianchi, & Chiaraluce, 2016).



Gambar 2.3 Algoritma McEliece

2.2.1 Pembangkitan Kunci

Untuk menghasilkan kunci publik yang dapat digunakan dalam proses enkripsi dan dekripsi diperlukan proses pembangkitan kunci yang terdiri atas beberapa langkah, berikut ini (Sinurat & Siagan, 2022):

1. Menentukan parameter-parameter dengan panjang (n), dimensi (k).
2. Membangkitkan matriks pengacak (S) berukuran $k \times k$ untuk mengacak elemen-elemen matriks generator.
3. Membangkitkan matriks generator (G) berukuran $k \times n$, dengan setiap elemen matriks yang didefinisikan dalam $GF(2^m)$.

4. Mengonstruksi matriks permutasi (M), dengan ukuran $n \times n$ untuk menambahkan *error* pada matriks generator.
5. Mengalikan matriks pengacak (S), matriks generator (G), dan matriks permutasi (M) untuk membentuk matriks kunci publik (G'), ditulis sebagai

$$G'_{k\times n} = S_{k\times k} \cdot G_{k\times n} \cdot M_{n\times n}$$

2.2.2 Enkripsi dan Dekripsi

Proses enkripsi pada algoritma McEliece dengan menggunakan kunci publik yang melibatkan matriks generator (G). Proses tersebut terdiri atas beberapa langkah, berikut ini (Sinurat & Siagan, 2022):

1. Mengonversi pesan asli (m) untuk menyelesaikan pesan dengan dimensi k blok.
2. Jika panjang pesan memenuhi dimensi k, pesan dapat ubah (*encoding*) menjadi kode biner. Jika panjang pesan kurang dari k, dilakukan penambahan $padding$ agar panjang pesan sesuai dengan dimensi k.
3. Menambahkan *error* (e) sebanyak n-bit.
4. Melakukan perkalian hasil *encoding* dengan kunci publik (G') yang telah dihasilkan dari pembangkitan kunci, yaitu

$$c_{n-bit} = m_{k-bit} \cdot G'_{k\times n} + e_{n-bit}$$

Proses dekripsi melibatkan proses *decoding* pada matriks generator untuk memastikan bahwa *error* yang ditambahkan dapat dikoreksi, sehingga pesan asli dapat dipulihkan kembali dengan kunci publik. Proses tersebut terdiri atas beberapa langkah, berikut ini (Sinurat & Siagan, 2022):

1. Agar pesan dapat dikembalikan ke bentuk sebelum ditambahkan *error*, dilakukan perkalian terhadap invers dari matriks permutasi (M), yaitu

$$c'_{n-bit} = c_{n-bit} \cdot M^{-1}_{n\times n}$$

2. Melakukan proses *decoding* terhadap *codeword* yang telah dikoreksi (c') menggunakan invers dari matriks generator (G'^{-1}).
3. Mengoreksi *error* yang mungkin terjadi selama transmisi atau enkripsi. Sehingga, dilakukan perkalian antara *codeword* yang telah dikoreksi dengan invers matriks generator, yaitu

$$m'_{n-bit} = c'_{k-bit} \cdot G'^{-1}_{k\times n}$$

2.3 Kode Hamming

Richard Hamming memperkenalkan kode Hamming pada tahun 1950, yang digunakan untuk mengoreksi dan mendeteksi kesalahan dalam blok bit. Selain itu, kode Hamming termasuk kategori kode linier, di mana setiap kode biner terdiri dari sejumlah bit data dan bit paritas yang ditambahkan ke data asli untuk memungkinkan deteksi dan koreksi kesalahan. Pada dasarnya, proses mengoreksi *error* (*error correction*) dan proses mendeteksi *error* (*error detection*) dilakukan kode Hamming menggunakan operasi logika XOR (*Exclusive OR*). Dalam penerapannya, data *input* dan *output* diubah menjadi bilangan biner yang terdiri dari bit-bit yang dapat berupa 0 atau 1 (Sembiring & Simbolon, 2021).

2.3.1 Pembangkitan Kunci

Pembangkitan kunci melibatkan penambahan bit paritas ke dalam data asli untuk memungkinkan deteksi dan koreksi kesalahan satu bit. Proses ini

menggunakan konsep kode *error correcting* dengan langkah-langkah, berikut ini (Hillier & Balyan, 2019):

1. Menentukan parameter-parameter dengan panjang (n), dan dimensi (k)
2. Membangkitkan matriks generator (G) berukuran $k \times n$, didefinisikan sebagai kombinasi matriks identitas (I) berukuran $k \times k$ dan sub-matriks paritas (P) berukuran $k \times (n-k)$, ditulis sebagai

$$G_{k\times n} = [I_{k\times k}|P_{k\times(n-k)}]$$

3. Membangkitkan matriks *parity-check* (H) berukuran $(n-k) \times n$, terdiri dari transpose sub-matriks paritas(P^T) berukuran $k \times (n-k)$ dan matriks identitas (I) berukuran $(n-k) \times (n-k)$, ditulis sebagai

$$H_{(n-k)\times n} = [P^T_{k\times(n-k)}|I_{(n-k)\times(n-k)}]$$

2.3.2 *Encoding* dan *Decoding*

Proses *encoding* pada kode Hamming melibatkan matriks generator (G) untuk mengkodekan pesan menjadi *codeword* yang siap dikirim. Proses tersebut terdiri atas beberapa langkah, berikut ini (Hillier & Balyan, 2019):

1. Menentukan vektor pesan asli (m) yang terdiri dari k-bit.
2. Menggunakan matriks generator (G) berukuran $k \times n$ yang telah dibangkitkan untuk menghasilkan *codeword* (c).
3. Melakukan perkalian vektor pesan (m) dengan matriks generator (G), yaitu

$$c_{n-bit} = m_{k-bit} \cdot G_{k\times n}$$

Proses *decoding* dilakukan untuk mendeteksi kesalahan yang mungkin terjadi selama transmisi dengan melibatkan matriks *parity-check* (H). Proses tersebut terdiri atas beberapa langkah, berikut ini (Hillier & Balyan, 2019):

1. Agar *codeword* (c) yang diterima dapat diperiksa kesalahan yang mungkin terjadi, dilakukan perkalian terhadap matriks *parity-check* (H), yaitu
$$S_{r-bit} = c_{n-bit} \cdot H^T_{r \times n}$$
2. Jika sindrom $S = 0$, maka tidak ada kesalahan, dan jika $S \neq 0$, posisi bit yang salah dapat dideteksi.
3. Memperbaiki kesalahan dilakukan pada posisi bit yang terdeteksi oleh sindrom S.

2.4 Otentikasi Dokumen

Otentikasi dokumen merupakan proses verifikasi integritas dan keaslian suatu dokumen yang dikirim atau diterima, terutama dalam saluran komunikasi digital seperti internet. Proses ini memastikan bahwa dokumen yang diterima benar-benar berasal dari sumber yang sah dan tidak mengalami perubahan atau manipulasi selama transmisi. Penerima dokumen dapat memverifikasi identitas pengirim dan memastikan bahwa isi dokumen tetap utuh sejak pertama kali dibuat. Selain itu, otentikasi juga berperan dalam meningkatkan kepercayaan antara pihak yang berkomunikasi secara digital dan mengurangi risiko penyalahgunaan informasi (Dlamini, Mthethwa, & Barbour, 2018).

Salah satu keunggulan dari otentikasi dokumen digital adalah peningkatan keamanan dan efisiensi dalam pertukaran dokumen. Dibandingkan dengan dokumen fisik, dokumen digital yang telah di otentikasi lebih sulit untuk dipalsukan

dan dapat diverifikasi secara otomatis menggunakan algoritma kriptografi. Berbagai metode digunakan dalam proses otentikasi ini, seperti kode QR (*Quick Response Code*), *watermarking*, dan *digital signature*, yang memungkinkan verifikasi dokumen dilakukan dengan cepat dan akurat. Namun, efektivitas otentikasi dokumen digital juga bergantung pada infrastruktur teknologi yang aman. Jika sistem yang digunakan mengalami kegagalan atau diretas, maka validitas dokumen dapat terancam. Oleh karena itu, diperlukan strategi yang tepat tanpa mengabaikan aspek keamanan dan kenyamanan pengguna (Sintyaningrum, Muladi, & Ashar, 2022).

2.4.1 Otentikasi Berbasis Kode QR (*Quick Response Code*)

Otentikasi dokumen berbasis kode QR merupakan salah satu metode verifikasi keaslian dokumen dengan menyematkan informasi unik dalam bentuk kode QR yang dapat dipindai menggunakan perangkat digital. Kode QR adalah kode batang 2 dimensi yang awalnya dirancang untuk industri otomotif Jepang oleh Denso Wave pada tahun 1994. Kemudian, pada tahun 2000 kode QR di standarisasi oleh *International Organization for Standardization* (ISO). Sistem kode QR menggunakan dua komponen, yaitu *encoding* dan *decoding* yang berperan dalam proses pembuatan dan pembacaan kode QR. *Encoding* berfungsi untuk menghasilkan kode QR yang terdiri dari pola hitam dan putih dalam bentuk matriks. Sementara itu, *decoding* berfungsi untuk memindai kode QR dan memverifikasi dokumen untuk memastikan keaslian dan menampilkan informasi yang terkandung di dalamnya (Wellem, Nataliani, & Iriani, 2022).

Tahapan yang dilakukan saat proses *encoding* untuk mengonversi dokumen menjadi kode QR dimulai dengan analisis dokumen, *encode* data, *error correction*, penambahan *data masking pattern*, dan pembangkitan kode QR. Selain itu, tahapan yang dilakukan saat proses *decoding* untuk memastikan dokumen dapat diterjemahkan dimulai dengan mendeteksi *quiet zone* dan *alignment pattern*, *decode* data, melakukan *error correction*, dan analisis dokumen. *Error correction* berfungsi agar kode QR tetap dapat dibaca meskipun terdapat bagian yang rusak atau terhapus (Lorien & Wellem, 2021).

2.4.2 Otentikasi dengan *Watermarking*

Otentikasi dokumen dengan metode *Watermarking* merupakan teknik perlindungan yang melibatkan penyisipan informasi tambahan pada dokumen yang tidak mengubah kontennya secara signifikan. *Watermarking* dapat bersifat *visible* (terlihat) atau *invisible* (tidak terlihat) tergantung pada tingkat keamanan yang diinginkan. *Visible watermarking* dapat berupa teks atau logo yang ditambahkan pada dokumen untuk menunjukkan kepemilikan atau status keasliannya. Sementara itu, *invisible watermarking* menggunakan teknik kriptografi untuk menyisipkan informasi yang hanya dapat diakses melalui metode tertentu, seperti dekripsi dengan kunci khusus (Bashardoost, Rahim, & Hadipour, 2015).

Watermarking memiliki berbagai aplikasi yang digunakan dalam berbagai bidang untuk melindungi dan mengamankan informasi digital. Aplikasi tersebut umumnya diklasifikasikan ke dalam beberapa kategori berdasarkan teknik *watermarking* yang digunakan dan tujuan penerapannya. Klasifikasi ini mencakup

watermarking untuk perlindungan hak cipta, di mana *watermarking* disisipkan guna mengidentifikasi kepemilikan dan mencegah pelanggaran hak cipta. Selain itu, terdapat *watermarking* untuk otentikasi dan deteksi manipulasi, yang berfungsi sebagai tanda tersembunyi dalam dokumen atau gambar untuk mendeteksi modifikasi yang tidak sah. Kategori lainnya meliputi *watermarking* untuk pengelolaan hak akses, di mana *watermarking* digunakan untuk mengatur izin penggunaan suatu konten digital, serta *watermarking* forensik yang membantu dalam pelacakan distribusi ilegal. Dengan adanya klasifikasi ini, teknik *watermarking* digital dapat diterapkan secara optimal sesuai dengan kebutuhan perlindungan dan keamanan informasi digital (Dixit & Dixit, 2017).

2.4.3 Otentikasi Berbasis *Digital Signature*

Otentikasi berbasis *digital signature* merupakan metode yang digunakan untuk memastikan keaslian dan integritas suatu dokumen digital dengan memanfaatkan prinsip kriptografi. *Digital signature* dibuat menggunakan *hashing algorithm*, di mana setiap perubahan sekecil pada dokumen akan menghasilkan *siganture* yang berbeda. Dengan cara ini, *digital signature* tidak hanya menjamin keaslian dokumen namun juga memastikan dokumen berasal dari sumber yang dapat dipercaya. Selain itu, teknologi ini mampu mencegah berbagai bentuk manipulasi dokumen, sehingga sering digunakan dalam transaksi elektronik dan komunikasi yang memerlukan tingkat keamanan tinggi. Dengan semakin meningkatnya kebutuhan dalam dunia digital, penggunaan *digital signature* menjadi standar baru dalam berbagai industri guna mendukung validitas dan keabsahan dokumen elektronik. (Lorien & Wellem, 2021).

2.5 *Digital Signature*

Digital signature atau tanda tangan digital merupakan penerapan kriptografi yang digunakan untuk mengotentikasi dan mengamankan dokumen secara digital. *Digital signature* berfungsi sebagai keamanan kriptografi yang bersifat *non-repudiation* atau anti penyangkalan. *Digital signature* bekerja dengan menggunakan sepasang kunci, yaitu kunci privat untuk menandatangani dokumen dan kunci publik untuk memverifikasi tanda tangan. Dalam prosesnya, dokumen yang akan di otentikasi terlebih dahulu dikonversi menjadi nilai *hash* menggunakan algoritma *hashing*, seperti SHA-3, guna memastikan ukuran data tetap konstan. Nilai *hash* yang dihasilkan kemudian dienkripsi menggunakan kunci privat pengirim, sehingga menghasilkan *digital signature* yang unik. Ketika dokumen diterima, penerima akan melakukan verifikasi dengan mendekripsi *digital signature* menggunakan kunci publik pengirim dan membandingkan hasilnya dengan nilai *hash* yang dihitung dari dokumen asli. Jika kedua nilai *hash* tersebut identik atau sama, maka dokumen dapat dipastikan keasliannya dan tidak mengalami perubahan sejak ditandatangani (Prabowo & Afrianto, 2017).

2.5.1 Fungsi *Hash* dalam *Digital Signature*

Fungsi *hash* adalah algoritma yang dirancang untuk mengonversi data masukan berupa *string* dengan panjang sembarang dan mengubahnya menjadi keluaran (*massage digest*) dengan panjang tetap (*fixed*). Fungsi ini bersifat satu arah (*one-way hash*), sehingga *massage digest* tidak dapat dikembalikan ke bentuk sebelum diubah untuk mendapatkan data asli dari nilai *hash*. Selain itu, *message digest* yang dihasilkan bersifat unik untuk setiap dokumen, sehingga jika terjadi

perubahan sekecil apa pun pada isi dokumen, maka nilai *hash* yang dihasilkan juga akan berubah secara signifikan (Harianja, 2023). Terdapat beberapa algoritma *hashing* yang umum digunakan dalam implementasi *digital signature*, diantaranya (Rahim dkk., 2023):

1. *Massage Digest* 5 (MD5)

   MD5 diperkenalkan oleh Profesor Ronald L. Rivest sebagai algoritma *hashing* yang menghasilkan *output* sepanjang 128-bit. Awalnya, algoritma ini menjadi pilihan populer dalam berbagai aplikasi kriptografi karena kecepatannya dalam memproses data dan menghasilkan nilai *hash* yang unik. Namun, seiring dengan perkembangan teknologi komputasi dan analisis keamanan, kelemahan dalam algoritma ini mulai terungkap. Salah satu celah keamanan terbesar MD5 adalah kerentanannya terhadap *collision attack*, yaitu kondisi di mana dua dokumen berbeda dapat menghasilkan nilai *hash* yang sama. Serangan ini memungkinkan untuk membuat dokumen palsu dengan *hash* yang identik dengan dokumen asli, sehingga dapat digunakan untuk manipulasi dokumen. Karena kelemahan ini, MD5 tidak lagi direkomendasikan untuk aplikasi keamanan yang memerlukan tingkat perlindungan tinggi.

2. *Secure Hash Algorithm* (SHA)

   SHA dikembangkan oleh *National Institute of Standards and Technology* (NIST) dan digunakan sebagai bagian dari *Digital Signature Standard* (DSS). Algoritma ini dianggap sebagai penerus MD5 dengan tingkat keamanan yang lebih tinggi, terutama dalam menghadapi serangan *collision attack*. Seiring perkembangannya, terdapat beberapa varian dari algoritma

SHA, di antaranya SHA-0, SHA-1, SHA-2, dan SHA-3, yang memiliki panjang *output* dan tingkat keamanan yang berbeda. SHA-1, yang sebelumnya banyak digunakan, kini dianggap rentan terhadap serangan dan telah digantikan oleh SHA-2 dan SHA-3 dalam banyak sistem keamanan modern. SHA-2 sendiri memiliki beberapa varian dengan panjang *hash* 224-bit, 256-bit, 384-bit, dan 512-bit, yang digunakan dalam enkripsi data, *digital signature*, dan autentikasi dalam sistem informasi. Sebagai varian terbaru, SHA-3 menawarkan desain yang lebih aman dibandingkan SHA-2 dan dirancang untuk mengatasi potensi kelemahan pada algoritma sebelumnya. Dengan fleksibilitas ukuran *output*, seperti 224-bit, 256-bit, 384-bit, dan 512-bit, SHA-3 menjadi pilihan utama dalam sistem keamanan tingkat tinggi yang termasuk dalam implementasi *post-quantum cryptography* yang dirancang untuk menghadapi ancaman dari komputasi kuantum.

2.5.2 Algoritma Kriptografi dalam *Digital Signature*

Proses mengamankan *message digest* yang telah diperoleh dilakukan dengan memanfaatkan algoritma kriptografi asimetris yang melibatkan dua kunci, yaitu kunci privat dan kunci publik. *Message digest* nantinya akan dienkripsi menggunakan kunci privat pengirim, sehingga tanda tangan hanya dapat diverifikasi oleh pihak yang memiliki kunci publik yang sesuai. Beberapa algoritma kriptografi asimetris yang umum digunakan dalam implementasi *digital signature* diantaranya (Suantara, Satwika, & Fredlina, 2024):

1. Rivest-Shamir-Adleman (RSA)

   RSA adalah salah satu algoritma kriptografi asimetris yang paling populer dan banyak digunakan dalam *digital signature*. Algoritma ini bekerja dengan memanfaatkan sifat matematika dari bilangan prima besar untuk menghasilkan pasangan kunci publik dan privat, yang digunakan dalam proses enkripsi dan dekripsi. Kunci publik digunakan untuk mengenkripsi pesan atau memverifikasi *digital signature*, sementara kunci privat digunakan untuk mendekripsi pesan atau membuat *digital signature* yang autentik.

2. *Digital Signature Algorithm* (DSA)

   DSA adalah algoritma yang dirancang khusus untuk menghasilkan *digital signature* dengan menggunakan konsep bilangan prima besar dan aritmetika modular. Algoritma ini dikembangkan oleh *National Institute of Standards and Technology* (NIST) sebagai bagian dari *Digital Signature Standard* (DSS). Salah satu keunggulan DSA adalah efisiensinya dalam proses pembuatan *digital signature*, yang lebih cepat dibandingkan dengan algoritma RSA. Namun, DSA memiliki keterbatasan dalam fleksibilitas penggunaan, karena hanya dapat digunakan untuk otentikasi dan *digital signature*, bukan untuk enkripsi pesan langsung.

3. *Elliptic Curve Digital Signature Algorithm* (ECDSA)

   ECDSA adalah varian dari DSA yang menggunakan prinsip kurva eliptik dalam proses pembangkitan kunci publik dan privat. ECDSA dapat menawarkan tingkat keamanan yang setara dengan algoritma RSA atau DSA, tetapi dengan ukuran kunci yang jauh lebih kecil. Misalnya, ECDSA

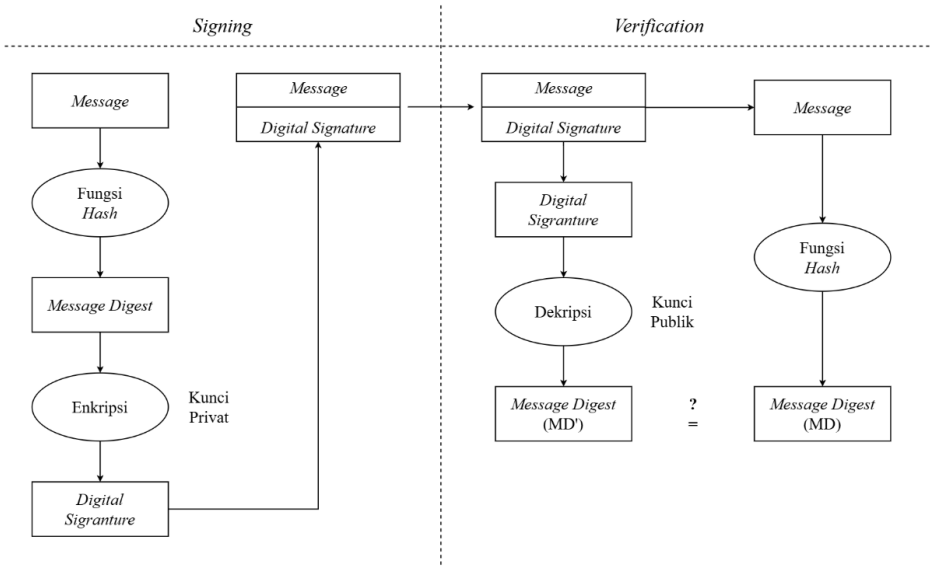
dengan kunci 256-bit memiliki tingkat keamanan yang sebanding dengan RSA 3072-bit, sehingga algoritma ini lebih efisien dalam penggunaan sumber daya komputasi dan penyimpanan.

4. McEliece

   Algoritma McEliece adalah sistem kriptografi berbasis kode yang menggunakan kode *error correcting* untuk enkripsi dan *digital signature*. Algoritma ini memanfaatkan matriks *parity-check* dalam pembangkitan kunci, di mana keamanan sistem bergantung pada kesulitan dalam memecahkan masalah *decoding* kode linier acak. Salah satu keunggulan utama McEliece adalah ketahanannya terhadap serangan komputer kuantum, menjadikannya salah satu kandidat algoritma *post-quantum cryptography* yang menjanjikan.

2.5.3 Proses *Signing* dan *Verifying*

Kombinasi fungsi *hash* dan algoritma kriptografi asimetris yang digunakan dalam proses *digital signature* digambarkan sebagai berikut :



Gambar 2.4 Skema *Digital Signature*

Seperti yang telah ditunjukkan pada Gambar 2.4, proses pemberian tanda tangan yang dilakukan oleh pengirim (*signing*), dengan langkah-langkah berikut:

1. Pengirim menggunakan fungsi *hash* untuk menghitung nilai *hash* dari pesan asli (m), sehingga diperoleh *message digest* (MD).
2. Pengirim mengenkripsi *message digest* (MD) dengan kunci privat, sehingga menghasilkan *digital signature*.
3. Pengirim mentransmisikan pesan asli (m) beserta *digital signature* yang diperoleh ke penerima.

Proses yang dilakukan oleh penerima untuk memeriksa atau verifikasi tanda tangan (*verifying*), dengan langkah-langkah berikut:

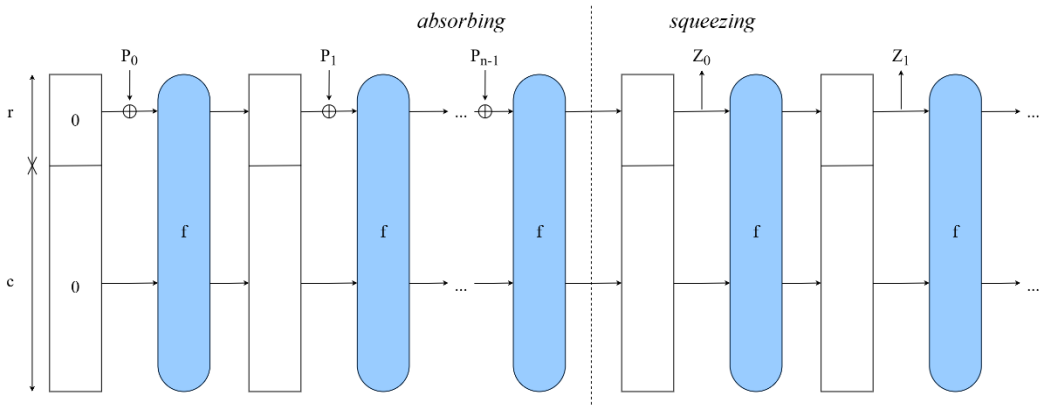
1. Penerima menggunakan fungsi *hash* untuk menghitung nilai *hash* dari pesan asli (m) yang diterima.
2. Pengirim mendekripsi *digital signature* menggunakan kunci publik pengirim, sehingga diperoleh *message digest* (MD').
3. Penerima membandingkan *message digest* awal (MD) dengan hasil deksipsi *message digest* (MD'). Jika $MD = MD'$ maka *digital signature* yang diperoleh otentik dan pesan yang diterima tidak mengalami perubahan ketika pengiriman.

2.6 Algoritma SHA-3 (*Secure Hash Algorithm*)

National Institute of Standards and Technology (NIST) merupakan laboratorium standar pengukuran teknologi yang berperan dalam pengelolaan dan pengurangan risiko serangan siber. Pada tahun, 2006, NIST mengadakan kompetisi

bernama *NIST Hash Function Competition* untuk menetapkan standar *hash* baru yang dikenal sebagai SHA-3. Pada tahun 2012, algoritma *Keccak* dinyatakan sebagai pemenang kompetisi setelah mengalahkan finalis lainnya, termasuk BLAKE, JH, Skein, dan Grøstl. Selanjutnya, pada tahun 2014, NIST mempublikasikan draft *Federal Information Processing Standards* (FIPS) 202 yang berjudul *"SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions"*. Kemudian, pada tahun 2015 diresmikan SHA-3 sebagai standar fungsi *hash* baru yang memiliki beberapa ukuran *output*, yaitu 224-bit, 256-bit, 384-bit, dan 512-bit (Romine, 2015).

SHA-3 dibuat tidak dikembangkan untuk menggantikan SHA-2, mengingat hingga saat ini belum terdapat serangan signifikan yang berhasil melemahkan algoritma SHA-2. Meskipun demikian, NIST memandang penting untuk menyediakan algoritma alternatif sebagai bentuk mitigasi terhadap potensi kerentanan di masa mendatang. Langkah ini diambil dengan mempertimbangkan sejarah kelemahan pada algoritma sebelumnya, seperti MD5, SHA-0, dan SHA-1, yang telah berhasil ditembus melalui berbagai teknik kriptanalisis. Oleh sebab itu, SHA-3 dibuat dengan tujuan untuk menghasilkan nilai *hash* yang unik dari data yang dimasukkan. Hal ini digunakan untuk berbagi tujuan keamanan data, seperti memastikan integritas data, membuat *checksum*, dan menyimpan sandi. Algoritma SHA-3 bekerja menggunakan konstruksi yang mirip dengan spons, dimulai dengan proses *absorbing* (penyerapan) dan dilanjutkan dengan *squeezing* (pemerasan). Secara sederhana fase tersebut digambarkan sebagaimana ditunjukkan pada Gambar 2.5.



Gambar 2.5 Konstruksi Spons SHA-3

Salah satu tahapan penting dalam pembuatan fungsi *hash* iteratif adalah fase *absorbing* (penyerapan), tahapan tersebut yaitu:

1. Pesan sebelumnya dikonversi menjadi bytes menggunakan UTF-8 dan diberi padding untuk dibagi menjadi beberapa blok (P_i) berukuran r-bit (rate).
2. Setiap blok input P_i di XOR dengan r-bit pertama dari state internal S.
3. Setelah operasi XOR, hasilnya diproses menggunakan fungsi permutasi f, untuk menghasilkan state baru.
4. Proses ini diulang untuk setiap blok *input* hingga semua blok selesai diproses.

Tahapan terakhir dalam proses konstruksi spons pada algoritma *hash* adalah fase *squeezing* (pemerasan), tahapan tersebut yaitu:

1. Variabel Z digunakan untuk menyimpan hasil akhir, yaitu *message digest* dan diinisialisasi sebagai *string* kosong.
2. Selama panjang Z belum mencapai panjang output yang diinginkan d (misalnya 256 bit untuk SHA3-256), mengambil r-bit pertama dari state S dan ditambahkan ke Z.

3. Jika panjang Z masih belum cukup, dilakukan permutasi lagi pada S menggunakan fungsi f, kemudian diulangi proses penambahan hingga Z mencapai panjang d.
4. Setelah cukup, Z menjadi hasil akhir *message digest*.

Dengan demikian, fase *squeezing* bertanggung jawab terhadap proses menghasilkan *message digest* atau nilai *hash* dari pesan input yang telah diproses selama fase *absorbing*. Nilai *hash* ini nantinya akan menjadi representasi singkat dari pesan input yang digunakan dalam memastikan integritas data atau dokumen, mengidentifikasi pesan.

2.7 Konversi Sistem Bilangan

Konversi bilangan merupakan proses mengubah suatu bilangan dari satu sistem bilangan ke sistem bilangan lainnya. Dalam dunia komputasi, konversi ini sangat penting karena berbagai sistem menggunakan basis bilangan yang berbeda untuk merepresentasikan data. Salah satu metode yang paling umum digunakan dalam konversi bilangan adalah metode "proses sisa". Metode ini melibatkan pembagian berturut-turut dari bilangan yang akan dikonversi dengan basis tujuan, di mana sisa hasil pembagian disimpan sebagai digit dalam sistem bilangan yang baru. Proses ini diulang hingga hasil pembagian menjadi nol, dan digit sisa yang diperoleh disusun dari bawah ke atas untuk mendapatkan hasil akhir. Terdapat empat jenis sistem bilangan yang digunakan untuk merepresentasikan nilai numerik dengan basis atau *radix* berbeda, yaitu (Supriyanto, 2020):

1. Sistem bilangan biner (*binary number system*), yaitu sistem yang menggunakan prinsip logika digital bekerja berdasarkan dua kondisi, yaitu *OFF* (tidak ada arus) dan *ON* (ada arus). Prinsip ini menjadi dasar sistem bilangan biner yang menggunakan 2 digit, yaitu 0 dan 1 yang digunakan untuk merepresentasikan nilai numerik.
2. Sistem bilangan oktal (*octal number system*), yaitu sistem yang menggunakan 8 digit (0 – 7) dan sering digunakan dalam aplikasi tertentu seperti sistem mikroprosesor dan pengkodean digital. Sistem ini memiliki hubungan erat dengan sistem biner karena setiap digit oktal dapat direpresentasikan oleh 3 digit biner, yang memudahkan konversi dan pemrosesan data dalam sistem komputer.
3. Sistem bilangan desimal (*decimal number system*), yaitu sistem yang paling sering digunakan oleh manusia karena menggunakan 10 digit (0 – 9) sebagai representasi numerik. Popularitas sistem ini didasarkan pada kebiasaan manusia menggunakan sepuluh jari sebagai alat bantu dalam perhitungan.
4. Sistem bilangan heksadesimal (*hexadecimal number system*), yaitu sistem yang menggunakan 16 digit, yaitu digit 0 – 9 dan huruf A – F yang mewakili nilai 10 – 15. Sistem ini sering digunakan dalam pemrograman komputer, khususnya dalam merepresentasikan warna dalam sistem digital, desain sirkuit, dan komunikasi data. Keunggulan sistem heksadesimal terletak pada kemampuannya merepresentasikan data biner dalam bentuk yang lebih ringkas dan mudah dibaca.

Sebagai contoh, sistem bilangan heksadesimal (basis 16) dapat dengan mudah dikonversi menjadi sistem bilangan lainnya seperti biner (basis 2), oktal (basis 8), dan desimal (basis 10). Konversi ini penting karena komputer bekerja menggunakan sistem biner untuk memproses informasi, sementara sistem oktal dan heksadesimal sering digunakan untuk menyederhanakan representasi bilangan biner yang panjang (Haryadi, 2023).

Tabel 2.1 Tabel Basis Bilangan

Biner	Oktal	Desimal	Heksadesimal
0 0 0 0	0	0	0
0 0 0 1	1	1	1
0 0 1 0	2	2	2
0 0 1 1	3	3	3
0 1 0 0	4	4	4
0 1 0 1	5	5	5
0 1 1 0	6	6	6
0 1 1 1	7	7	7
1 0 0 0	10	8	8
1 0 0 1	11	9	9
1 0 1 0	12	10	a
1 0 1 1	13	11	b
1 1 0 0	14	12	c
1 1 0 1	15	13	d
1 1 1 0	16	14	e
1 1 1 1	17	15	f

2.7.1 Konversi Bilangan Heksadesimal ke Biner

Salah satu bentuk konversi yang sering digunakan dalam bidang komputasi adalah konversi bilangan heksadesimal ke bilangan biner. Proses ini penting karena bilangan biner merupakan bahasa dasar yang digunakan oleh komputer

dalam memproses data. Oleh karena itu, dalam penelitian ini dipilih sistem bilangan heksadesimal dan biner yang memiliki peran penting dalam proses enkripsi dan dekripsi menggunakan algoritma SHA-3 dan McEliece. Pada proses *hashing*, algoritma SHA-3 mengonversi data masukan menjadi nilai *hash* dalam bentuk heksadesimal Setiap digit pada bilangan heksadesimal dapat direpresentasikan ke dalam bilangan biner 4-bit. Representasi ini kemudian diolah lebih lanjut dalam proses enkripsi menggunakan algoritma McEliece, yang memanfaatkan matriks generator dan matriks pemeriksaan paritas (Haryadi, 2023).

Sebagai contoh sederhana, misalkan suatu dokumen memiliki nilai *hash* $67bc28$ dan diperoleh bilangan biner dari konversi sebagai berikut:

$\underline{6}$	$\underline{7}$	$\underline{b}$	$\underline{c}$	$\underline{2}$	$\underline{8}$
↓	↓	↓	↓	↓	↓
0 1 1 0	0 1 1 1	1 0 1 1	1 1 0 0	0 0 1 0	1 0 0 0

Sehingga, diperoleh

$$67bc28 = 0\,1\,1\,0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0$$

Berdasarkan penjelasan di atas, metode ini dapat digunakan untuk mengonversi bilangan heksadesimal apa pun menjadi bilangan biner. Proses konversi dilakukan dengan mengubah masing-masing digit heksadesimal menjadi representasi biner 4-bit yang sesuai, berdasarkan sistem bilangan berbasis dua. Setiap digit dikonversi secara terpisah tanpa mempertimbangkan posisi desimalnya, karena konversi dari heksadesimal ke biner bersifat langsung dan tidak memerlukan perhitungan berdasarkan pangkat dua seperti pada konversi desimal ke biner.

2.8 Analisis Keamanan dan Pengujian Waktu Pemrosesan

2.8.1 Analisis Keamanan sebagai Evaluasi Algoritma

Analisis keamanan merupakan salah satu komponen penting dalam evaluasi algoritma kriptografi. Aspek keamanan tidak hanya berkaitan dengan proses enkripsi atau dekripsi, tetapi mencakup ketahanan algoritma terhadap berbagai jenis serangan, seperti *brute force*, *differential attack*, dan *side-channel attack*. Salah satu tolok ukur utama dalam menguji tingkat keamanan yaitu dengan melakukan pengujian terhadap kemampuan algoritma dalam mengacak kombinasi antara *input* dan *output*. Jika algoritma berhasil membuat perubahan *output* yang signifikan meskipun hanya terjadi perubahan kecil pada *input*, maka tingkat keamanannya dianggap tinggi karena sulit dilakukan rekonstruksi data asli. Oleh karena itu, pengujian terhadap prediksi *input* menjadi indikator penting dalam menilai seberapa kuat suatu algoritma dalam menjaga kerahasiaan data (Kurniawan, Kusyanti, & Nurwarsito, 2017). Salah satu metode pengujian keamanan algoritma diantaranya:

1. Efek Avalanche, metode ini digunakan untuk mengukur seberapa besar perubahan *output* yang terjadi akibat perubahan pada *input*. Algoritma yang baik akan menghasilkan perubahan *output* sebesar 50% dari total bit, menandakan bahwa algoritma memiliki difusi yang kuat dan sulit diprediksi (Upadhyay, Gaikwad, Zaman, & Sampalli, 2022). Proses perhitungan dilakukan dengan membandingkan dua *output hashing*, yaitu sebelum dan sesudah adanya perubahan satu bit pada *input* menggunakan persamaan berikut:

$$Avalanche\ Effect\ (\%) = \left(\frac{jumlah\ bit\ yang\ berubah}{total\ bit\ output}\right) \times 100\%$$

2. Distribusi bit *output*, metode ini digunakan untuk memastikan bahwa hasil enkripsi atau *hashing* menyebar secara acak dan merata. Distribusi yang tidak merata dapat menunjukkan kelemahan algoritma karena membuka celah pola tertentu yang bisa dimanfaatkan dalam serangan kriptanalisis. Evaluasi distribusi bit *output* dapat dilakukan dengan menganalisis frekuensi kemunculan bit 0 dan 1 pada *output*. Pengujian dapat dilakukan dengan uji statistik seperti uji chi-kuadrat untuk membandingkan distribusi aktual dengan distribusi yang diharapkan.
3. *Character error rate*, metode ini digunakan untuk mengukur tingkat kesalahan karakter antara teks asli dan hasil proses, terutama dalam sistem pengenalan ucapan dan optik karakter. Perhitungan CER dilakukan dengan membandingkan teks hasil dengan teks referensi menggunakan persamaan berikut:

$$CER = \frac{S + D + I}{N}$$

di mana S adalah julah substitusi, D adalah jumlah penghapusan (*deletion*), I adalah jumlah penyisipan (*insertion*), dan N adalah jumlah total karakter dalam teks referensi. Nilai CER yang tinggi pada proses kriptografi dapat menunjukkan distorsi yang baik untuk keamanan, namun juga harus dijaga agar tidak merusak integritas pesan asli (Sawata, Kashiwagi, & Takahashi, 2022).

Pada penelitian ini, pengujian menggunakan Efek Avalanche karena lebih efektif menunjukkan sensitivitas algoritma terhadap perubahan kecil pada *input*. Efek ini memberikan gambaran seberapa besar perubahan *output* yang terjadi

akibat perubahan satu bit pada *input* dalam waktu singkat. Sehingga, pengujian ini lebih tepat dalam mengukur keamanan dan pengacakan algoritma dibandingkan distribusi bit *output* atau *character error rate*.

2.8.2 Pengujian Waktu Pemrosesan dalam Keamanan Digital

Waktu pemrosesan merujuk pada durasi yang dibutuhkan sistem untuk menyelesaikan fungsi-fungsi utama seperti enkripsi, *hashing*, maupun proses verifikasi *digital signature*. Waktu ini menjadi indikator penting dalam menilai efisiensi suatu algoritma, terutama ketika diterapkan dalam sistem berskala besar atau *real-time*. Efisiensi waktu menjadi sangat krusial dalam beberapa konteks seperti *Internet of Things* (IoT), perangkat *mobile*, dan lingkungan komputasi awan (*cloud computing*), di mana keterbatasan daya komputasi, *bandwidth*, dan konsumsi energi menjadi tantangan utama. Sistem kriptografi yang ringan namun tetap aman sangat dibutuhkan agar tidak membebani perangkat dan memastikan keamanan data tetap terjaga tanpa mengorbankan waktu respons yang dibutuhkan pengguna (Varela, Ospino, & Lezama, 2020).

Faktor yang memengaruhi waktu pemrosesan antara lain kompleksitas algoritma yang digunakan, di mana semakin rumit struktur dan langkah-langkah dalam algoritma, maka semakin lama proses yang dibutuhkan. Selain itu, data berukuran besar tentu memerlukan waktu lebih lama dibandingkan data yang lebih kecil. Spesifikasi perangkat keras yang digunakan, seperti kecepatan prosesor dan kapasitas memori, turut menentukan cepat atau lambatnya proses eksekusi algoritma. Oleh karena itu, dalam pengembangan algoritma kriptografi keseimbangan antara tingkat keamanan dan efisiensi pemrosesan sanggat penting

agar sistem dapat tetap aman tanpa mengorbankan efektivitasnya dalam kondisi operasional.

2.9 Kajian Islam

Islam mengajarkan umatnya nilai-nilai kejujuran, keadilan, dan amanah dalam setiap interaksi sosial. Kejujuran tercermin dalam ketulusan hati yang dijaga dalam setiap ucapan dan tindakan, terutama dalam berjanji atau bertransaksi. Keadilan berarti memperlakukan hak-hak dengan setara dan sesuai ketentuan. Sementara itu, prinsip amanah menekankan pada pentingnya kepercayaan dan tanggung jawab dalam hubungan. Dalam dunia digital, prinsip-prinsip ini diimplementasikan melalui sistem keamanan informasi yang bertujuan untuk menjaga integritas, kerahasiaan, dan keaslian dokumen (Puspitasari, 2023). Bentuk nyata dari implementasi nilai kejujuran, keadilan, dan amanah dapat diwujudkan melalui mekanisme otentikasi seperti *digital signature*. *Digital signature* atau tanda tangan digital merupakan jaminan modern atas keaslian dokumen dan identitas pengirim. Teknologi ini mencerminkan prinsip kejujuran, keadilan, dan amanah dalam komunikasi digital, sekaligus menjadi sarana penerapan etika Islam di era modern.

Nilai otentikasi ini sejalan dengan ajaran Islam yang mengutamakan pentingnya mencatat, memverifikasi, dan menjaga amanah dalam setiap perjanjian atau dokumen. Islam mengajarkan agar setiap transaksi dan kesepakatan tercatat dengan jelas, dapat dipertanggungjawabkan, dan tidak disalahgunakan. Oleh karena itu, mekanisme otentikasi *digital signature* bukan hanya relevan dalam konteks teknologi, tetapi juga mencerminkan prinsip-prinsip Islam tentang kejujuran dan

integritas dalam menjaga amanah. Oleh karenanya, ditegaskan dalam Al-Qur'an Surat Al-Baqarah ayat 283 yang berbunyi (Kementrian Agama, 2022c):

وَاِنْ كُنْتُمْ عَلٰى سَفَرٍ وَّلَمْ تَجِدُوْا كَاتِبًا فَرِهٰنٌ مَّقْبُوْضَةٌۗ فَاِنْ اَمِنَ بَعْضُكُمْ بَعْضًا فَلْيُؤَدِّ الَّذِى اؤْتُمِنَ اَمَانَتَه
وَلْيَتَّقِ اللّٰهَ رَبَّهٗۗ وَلَا تَكْتُمُوا الشَّهَادَةَۗ وَمَنْ يَّكْتُمْهَا فَاِنَّهٗٓ اٰثِمٌ قَلْبُهٗۗ وَاللّٰهُ بِمَا تَعْمَلُوْنَ عَلِيْمٌ

Artinya: "Jika kamu dalam perjalanan, sedangkan kamu tidak mendapatkan seorang pencatat, hendaklah ada barang jaminan yang dipegang. Akan tetapi, jika sebagian kamu memercayai sebagian yang lain, hendaklah yang dipercayai itu menunaikan amanatnya (utangnya) dan hendaklah dia bertakwa kepada Allah, Tuhannya. Janganlah kamu menyembunyikan kesaksian karena siapa yang menyembunyikannya, sesungguhnya hatinya berdosa. Allah Maha Mengetahui apa yang kamu kerjakan." (QS. Al-Baqarah: 283)

Ayat ini mewajibkan mencatat transaksi dan pentingnya memiliki bukti atau jaminan tertulis untuk setiap bentuk perjanjian. Ayat ini menunjukkan betapa Islam sangat mendorong transparansi, kejelasan, dan tanggung jawab dalam setiap akad atau muamalah agar tidak timbul perselisihan atau ketidakadilan di kemudian hari. Proses pencatatan bukti menjadi salah satu bentuk proteksi terhadap hak dan kewajiban dari masing-masing pihak, terutama dalam kondisi yang rawan penipuan atau kelalaian (Az-Zuhaili, 2016).

Ajaran dalam ayat ini memberikan dasar normatif bagi pentingnya sistem otentikasi dalam menjaga keabsahan transaksi, termasuk dalam bentuk digital sekalipun. Sejalan dengan prinsip ini, nilai-nilai Islam tidak hanya menekankan pada aspek keabsahan suatu transaksi, tetapi juga pada prinsip amanah dan keadilan dalam pengambilan keputusan atau penetapan hukum. Sehingga, *Digital signature* mendukung prinsip ini dengan memastikan keaslian dokumen, mencegah pemalsuan, dan menjamin dokumen diterima oleh pihak yang berhak. Konsep ini telah tertulis dalam Al-Qur'an surat An-Nisa' ayat 58 yang berbunyi (Kementerian Agama, 2022d):

اِنَّ اللّٰهَ يَأْمُرُكُمْ اَنْ تُؤَدُّوا الْاَمٰنٰتِ اِلٰٓى اَهْلِهَاۙ وَاِذَا حَكَمْتُمْ بَيْنَ النَّاسِ اَنْ تَحْكُمُوْا بِالْعَدْلِۗ اِنَّ اللّٰهَ نِعِمَّا يَعِظُكُمْ بِهٖۗ اِنَّ اللّٰهَ كَانَ سَمِيْعًاۢ بَصِيْرًا

Artinya: "Sesungguhnya Allah menyuruh kamu menyampaikan amanah kepada pemiliknya. Apabila kamu menetapkan hukum di antara manusia, hendaklah kamu tetapkan secara adil. Sesungguhnya Allah memberi pengajaran yang paling baik kepadamu. Sesungguhnya Allah Maha Mendengar lagi Maha Melihat." (QS. An-Nisa': 58)

Ayat ini mengajarkan betapa pentingnya menjaga amanah dalam segala aspek kehidupan, termasuk dalam menjaga kerahasiaan informasi yang telah dipercayakan kepada seseorang. Allah memerintahkan agar setiap amanah disampaikan dengan penuh tanggung jawab, tepat sasaran, dan diberikan kepada mereka yang berhak menerimanya. Selain itu, ayat ini juga menegaskan pentingnya berlaku adil dalam menetapkan hukum di antara manusia, tanpa keberpihakan dan berdasarkan kebenaran yang hakiki (Az-Zuhaili, 2016).

Penerapan teknologi *digital signature* tidak hanya memenuhi aspek keamanan teknis, tetapi juga mencerminkan kepatuhan terhadap nilai-nilai syariat Islam dalam menjaga kepercayaan, keadilan, dan tanggung jawab dalam setiap proses pengambilan keputusan. Oleh karena itu, implementasi *digital signature* dapat dianggap sebagai bentuk ikhtiar manusia dalam menjaga keabsahan dan kebenaran dokumen. Hal ini, juga dikuatkan melalui penggunaan algoritma SHA-3 dan McEliece yang menjadi alat untuk mewujudkannya. Sehingga, setiap transaksi atau kesepakatan yang dilakukan tetap berada dalam koridor nilai-nilai keislaman yang menjunjung tinggi keadilan dan tanggung jawab. Selain itu, penerapan konsep ini menunjukkan bahwa agama dan teknologi menciptakan keseimbangan antara etika dan inovasi, baik dengan landasan moral yang kuat maupun pemanfaatan ilmu pengetahuan yang bijak.

BAB III

METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini menggunakan pendekatan kualitatif, yang berfokus pada implementasi proses otentikasi dokumen digital yang bersifat deskriptif. Data penelitian dianalisis dalam bentuk dokumen digital yang memuat alfabet, angka, dan karakter yang diperoleh dari dokumen pribadi penulis.

3.2 Tahapan Penelitian

Tahapan dalam proses verifikasi dokumen menggunakan algoritma SHA-3 dan McEliece dengan kode Hamming berbasis *digital signature* membutuhkan parameter (n, k), di mana n adalah panjang kolom matriks dan k adalah dimensi baris matriks. Nilai n dan k merupakan bilangan relatif prima, yaitu bilangan yang tidak memiliki faktor pembagi selain 1. Selain itu, panjang kolom dan dimensi baris terdiri dari nilai biner 0 dan 1, di mana elemen-elemen tersebut merepresentasikan struktur matriks dalam algoritma McEliece yang digunakan. Seluruh proses dilakukan menggunakan program dengan bahasa pemrograman Python untuk menjamin efisiensi, fleksibilitas, dan integrasi dengan berbagai pustaka kriptografi modern.

3.2.1 Pembangkitan Kunci

Kunci yang diperlukan pada proses *signing* dan *verifying* dalam otentikasi dokumen menggunakan algoritma SHA-3 dan McEliece melibatkan beberapa langkah berikut ini (Alahmadi dkk., 2023):

1. Membangkitkan matriks generator (G) berukuran $k \times n$ secara acak.
2. Menghitung matriks *parity-check* (H) dari matriks generator (G) berukuran $(n-k) \times n$.
3. Membangkitkan matriks pengacak (S) berukuran $(n-k) \times (n-k)$ secara acak.
4. Pemilihan vektor *noise* non-nol h, berukuran $n-k$.
5. Perkalian transpose dari matriks *parity-check* (H) dengan matriks pengacak (S) untuk membentuk matriks kunci privat (H'), ditulis sebagai

$$H'_{n\times(n-k)} = H^T_{n\times(n-k)} \cdot S_{(n-k)\times(n-k)}$$

3.2.2 *Input* Dokumen

Proses *input* dokumen dilakukan sistem untuk membaca isi dokumen yang akan dikirimkan. Dokumen yang di *input* harus memenuhi syarat yaitu dokumen berformat .pdf yang memuat alfabet, angka, dan karakter.

3.2.3 Proses *Hashing*

Proses *hashing* dilakukan untuk membentuk nilai *hash* dari dokumen menggunakan algoritma SHA-3. Proses ini menghasilkan *message digest* dimulai dengan fase *absorbing*, sebagai berikut:

1. Pesan dikonversi menggunakan UTF-8 dan diberi padding untuk dibagi menjadi blok berukuran r-bit, di mana $r = 1088$ bit $= 136$ byte.
2. Setiap blok di XOR dengan panjang 136 byte pertama dari state S, di mana $S = 1600$ bit $= 200$ byte.

Selanjutnya, fase *squeezing* untuk membentuk *message digest,* sebagai berikut:

1. Setelah semua blok diproses, dilakukan pengambilan *output* dari 136 byte pertama dari *state* dan ditambahkan ke Z.
2. Karena $256 \text{ bit} = 32 \text{ byte}$, dan $32 \text{ byte} < 136 \text{ byte}$, maka cukup satu iterasi untuk menghasilkan *message digest*.

3.2.4 Konversi *Message Digest*

Agar *Message digest* yang diperoleh dapat digunakan dalam tahap selanjutnya, dilakukan langkah berikut ini:

1. Setiap *message digest* yang awalnya dalam format heksadesimal dikonversi menjadi format biner.
2. Setiap urutan biner yang diperoleh dibagi menjadi blok-blok yang terdiri dari n-bit, yang disebut dengan vektor pesan (m).
3. Jika panjang bit tidak mencapai n-bit, maka akan dilakukan *padding* dengan menambah 0 pada bit terakhir agar struktur data tidak mengubah *digital signature* yang diperoleh.

3.2.5 *Signing* Dokumen

Proses penandatanganan dokumen menggunakan *digital signature* dilakukan melalui beberapa langkah berikut ini (Alahmadi dkk., 2023):

1. Vektor pesan (m) dengan ukuran n-bit.
2. Melakukan perkalian vektor pesan (m) dengan matriks kunci privat (H') untuk memperoleh *ciphertext* (c), yang disebut *digital signature* yaitu

$$c_{(n-k)-bit} = m_{n-bit} \cdot H'_{n\times(n-k)} + h_{(n-k)-bit}$$

3. Mengirimkan dokumen berupa pasangan dari dokumen asli dan *ciphertext* (c) yang telah ditandatangani.

3.2.6 *Verifying* Dokumen

Proses verifikasi dokumen menggunakan *digital signature* dilakukan melalui beberapa langkah berikut ini (Alahmadi dkk., 2023):

1. Melakukan perkalian ciphertext (c) dari *digital signature* dengan invers dari matriks pengacak (S^{-1}) untuk memperoleh dekripsi *ciphertext* (c'),

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

2. Melakukan *decoding* pada dokumen untuk memperoleh pesan vektor (m). Jika hasil *decoding* pesan vektor (m) sama dengan dekripsi *ciphertext* (c'), maka dokumen dianggap valid.

$$c' = cS^{-1} = (m \cdot H' + h)S^{-1} = mH'S^{-1} + hS^{-1}$$

$$\Rightarrow cS^{-1} = mH^{T}SS^{-1} + hS^{-1}$$

$$\Rightarrow c_{(n-k)-bit}S^{-1}_{(n-k)\times(n-k)} = m_{n-bit}H^{T}_{n\times(n-k)} + h_{(n-k)-bit}S^{-1}_{(n-k)\times(n-k)}$$

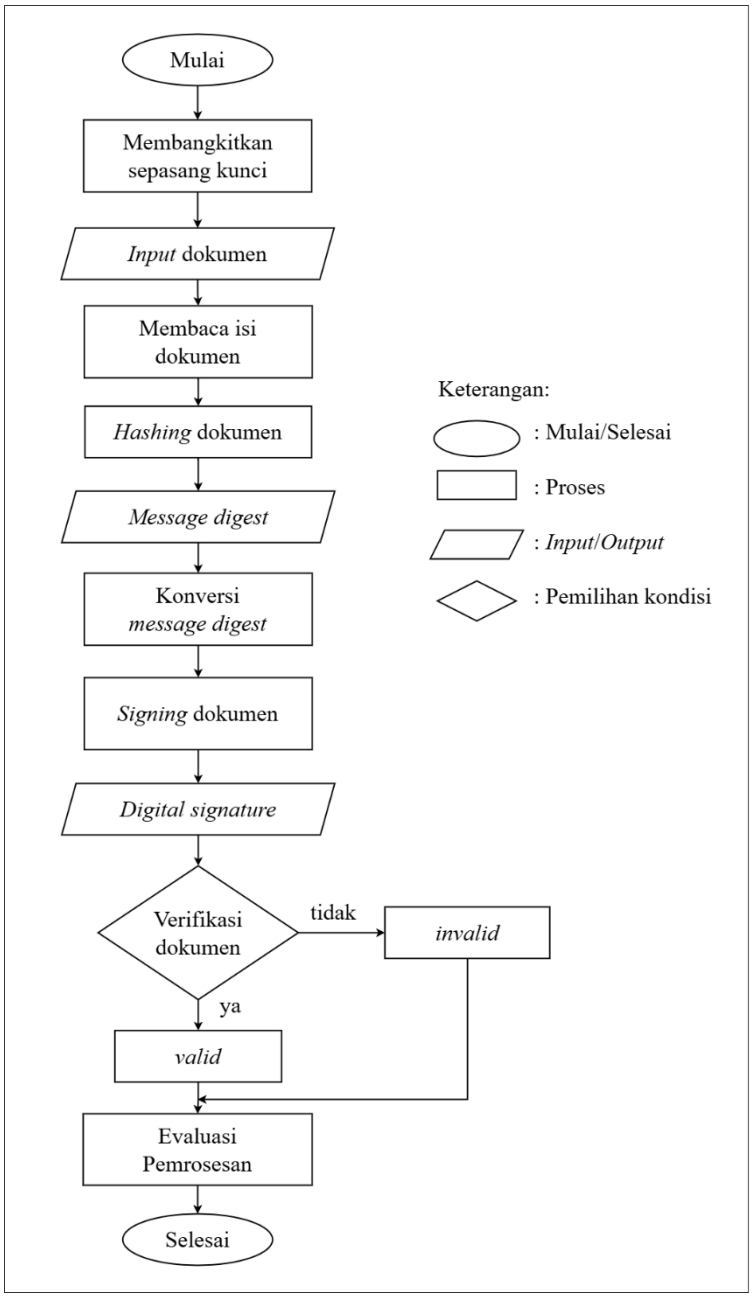
3.2.7 Evaluasi Keamanan dan Efisiensi Pemrosesan

Tahapan terakhir dalam penelitian ini yaitu melakukan evaluasi terhadap algoritma yang telah di tentukan menggunakan program Python, sebagai berikut:

1. Melakukan analisis keamanan menggunakan metode Efek Avalanche, dengan mencari persentase perubahan bit pada *output*.
2. Melakukan pengujian waktu pemrosesan dengan mengukur durasi yang dibutuhkan dalam dua pendekatan algoritma SHA-3, yaitu dengan dan tanpa algoritma McEliece sebagai mekanisme tambahan.

3.3 *Flowchart*

Flowchart pada Gambar 3.1 berikut menunjukkan alur kerja dari sistem otentikasi dokumen berbasis *digital signature* yang diimplementasikan menggunakan algoritma SHA-3 dan McEliece dengan kode Hamming.



Gambar 3.1 *Flowchart* Proses Otentikasi Dokumen

BAB IV

HASIL DAN PEMBAHASAN

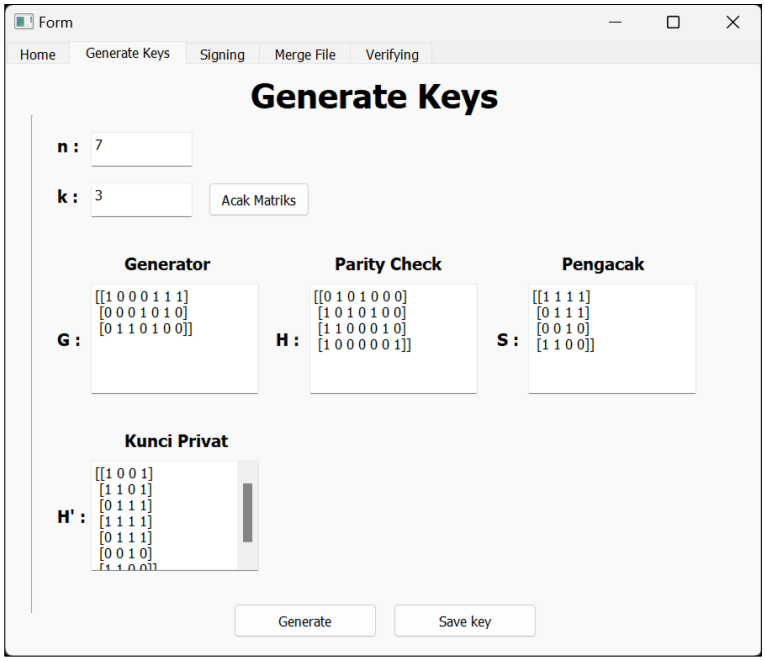
4.1 Pembangkitan Kunci

Proses *signing* dan *verifying* dokumen tidak terlepas dari proses pembangkitan kunci yang berperan penting dalam menjamin keamanan dan integritas *digital signature*. Dalam implementasi algoritma SHA-3 dan McEliece, proses pembangkitan kunci melibatkan tiga jenis matriks yang akan digunakan. Matriks pertama yaitu matriks generator (G), matriks ini dibangkitkan secara acak terdiri dari nilai biner 0 dan 1, di mana elemen-elemen tersebut merepresentasikan struktur matriks dalam algoritma McEliece yang berfungsi sebagai dasar pembentukan kode linier untuk merepresentasikan pesan asli. Selanjutnya, terdapat Matriks *parity-check* (H) yang digunakan untuk memverifikasi integritas dokumen selama transmisi atau penyimpanan, yang dibentuk dari kombinasi matriks generator dan matriks identitas. Matriks ketiga yaitu matriks pengacak (S) yang juga dibangkitkan secara acak dan digunakan menyamarkan kode selama proses enkripsi dan dekripsi agar tidak terkait langsung dengan struktur data asli. Ketiga matriks tersebut membentuk kunci publik, yang dapat dibagikan secara terbuka untuk keperluan verifikasi *digital signature*. Sementara itu, proses *signing* menggunakan kunci privat (H') yang dibentuk dari kombinasi matriks lainnya dan hanya diketahui oleh pihak yang berwenang. Ditambahkan juga noise acak h pada *ciphertext* untuk meningkatkan keamanan, sehingga pesan terenkripsi menjadi lebih sulit dianalisis atau diprediksi oleh pihak yang tidak berwenang. Dengan sistem ini, setiap *digital signature* yang dihasilkan bersifat unik, sulit dipalsukan, dan dapat diverifikasi secara akurat.

Pada penelitian ini akan dibangkitkan dua kunci berbeda guna mengevaluasi sejauh mana pengaruh perubahan *digital signature* yang diperoleh selama proses *signing* dan *verifying* dokumen. Proses ini bertujuan untuk mengamati konsistensi serta sensitivitas sistem terhadap perubahan kunci, yang berperan penting dalam aspek keandalan dan keamanan sistem *digital signature*. Seluruh proses dilakukan menggunakan program dengan bahasa pemrograman Python untuk menjamin efisiensi, fleksibilitas, dan integrasi dengan berbagai pustaka kriptografi modern. Untuk mendukung kenyamanan dan kemudahan penggunaan, tampilan antarmuka sistem dirancang menggunakan PyQt5. Hal ini bermaksud memungkinkan pengguna berinteraksi dengan sistem secara visual, mulai dari menentukan parameter yang akan digunakan hingga membentuk kunci publik dan kunci privat. Dengan demikian, sistem tidak hanya mendukung aspek keamanan data, tetapi juga memperhatikan aspek *user interface* dalam proses penandatangan dokumen digital.

4.1.1 Kunci Pertama

Pengujian pertama dilakukan pada dokumen dengan matriks kunci yang akan digunakan dalam proses *signing* dan *verifying* dokumen. Adapun matriks kunci yang digunakan dalam pengujian ini menggunakan parameter (7,3) yang dibangkitkan dengan bantuan program Python. Parameter tersebut menunjukkan panjang kolom dan dimensi baris yang terdiri dari nilai biner 0 dan 1, di mana elemen-elemen tersebut merepresentasikan struktur matriks dalam algoritma McEliece yang digunakan. Dengan parameter ini, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.1.



Gambar 4.1 Pembangkitan Kunci Pertama

Berdasarkan Gambar 4.1, matriks kunci publik dan kunci privat tersebut dapat ditulis, sebagai berikut:

1. Matriks generator (G) berukuran $k \times n$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

2. Matriks *parity-check* (H) dari G berukuran $(n-k) \times n$

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Matriks pengacak (S) berukuran $(n-k) \times (n-k)$

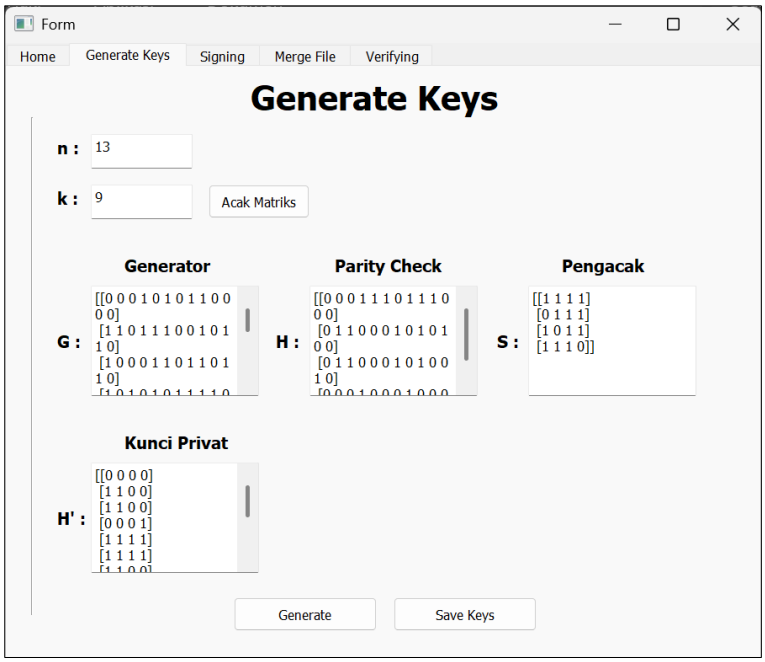
$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

4. Matriks kunci privat (H') berukuran $(n \times (n-k))$

$$H' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

4.1.2 Kunci Kedua

Pengujian kedua dilakukan pada dokumen yang sama namun menggunakan matriks kunci yang berbeda. Perbedaan matriks kunci ini bertujuan untuk mengevaluasi pengaruh perubahan kunci terhadap proses *signing* dan *verifying* dokumen. Adapun matriks kunci yang digunakan dalam pengujian ini menggunakan parameter $(13,9)$ yang dibangkitkan dengan bantuan program Python. Dengan parameter ini, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.2.



Gambar 4.2 Pembangkitan Kunci Kedua

Berdasarkan Gambar 4.2, matriks kunci publik dan kunci privat tersebut dapat ditulis, sebagai berikut:

1. Matriks generator (G) berukuran $k \times n$

$$G = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

2. Matriks *parity-check* (H) dari G berukuran $(n-k) \times n$

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Matriks pengacak (S) berukuran $(n-k) \times (n-k)$

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

4. Matriks kunci privat (H') berukuran $(n \times (n-k))$

$$H' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

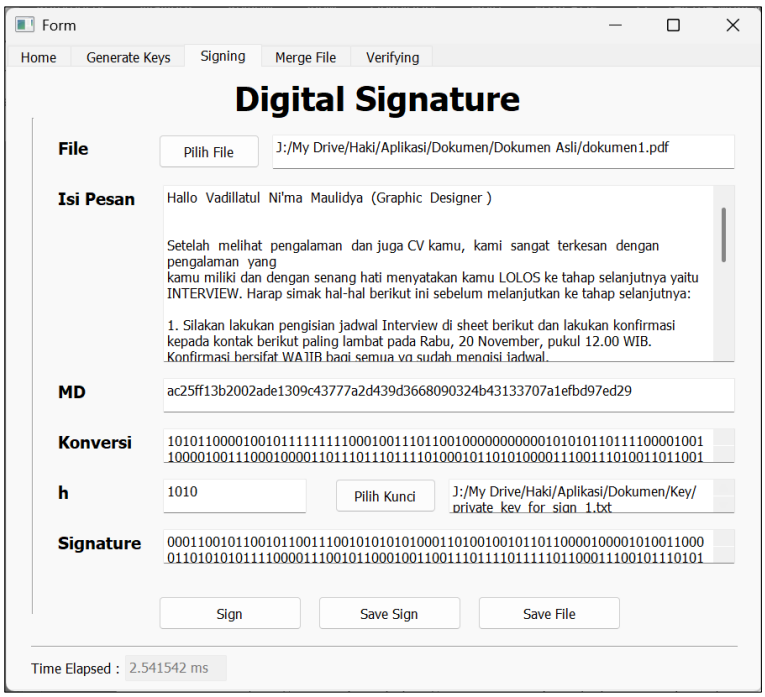
4.2 *Signing* Dokumen

Proses penandatanganan dokumen dilakukan dengan mengubah isi dokumen menjadi representasi digital melalui proses *hashing* menggunakan algoritma SHA-3 yang disebut *message digest*. Selanjutnya, *message digest* yang dihasilkan akan dikonversi menjadi bentuk biner agar dapat diproses lebih lanjut pada tahapan enkripsi. Tahapan enkripsi dilakukan menggunakan algoritma McEliece yang telah terbukti memiliki keamanan tingkat tinggi terhadap serangan komputer kuantum. Hasil dari tahapan enkripsi *message digest* adalah *digital signature* yang bersifat unik untuk setiap dokumen dan kunci yang digunakan. Sementara itu, untuk memudahkan dalam proses verifikasi hasil penandatanganan dokumen akan diubah menjadi QR Code, sehingga hasil *digital signature* dapat dengan mudah dibaca, dibagikan, dan diverifikasi oleh sistem penerima. Perubahan ini juga memungkinkan penyimpanan dan transmisi tanda tangan digital dalam bentuk yang lebih ringkas dan praktis.

Metode ini diterapkan pada dua puluh dokumen berformat .pdf, berisi alfabet, angka, dan karakter. Seluruh proses dilakukan menggunakan program dengan bahasa pemrograman Python untuk menjamin efisiensi, fleksibilitas, dan integrasi dengan berbagai pustaka kriptografi modern. Untuk mendukung kenyamanan dan kemudahan penggunaan, tampilan antarmuka sistem dirancang menggunakan PyQt5. Hal ini bermaksud memungkinkan pengguna berinteraksi dengan sistem secara visual, mulai dari pemilihan dokumen, proses *hashing*, pembentukan *digital signature*, hingga pembangkitan QR Code untuk keperluan verifikasi. Dengan demikian, sistem tidak hanya mendukung aspek keamanan data, tetapi juga memperhatikan aspek *user interface* dalam proses penandatangan dokumen digital.

4.2.1 Dokumen Asli dengan Kunci Pertama

Pengujian pertama dilakukan pada dokumen satu halaman dengan nama *dokumen1.pdf*, sebagaimana tercantum pada Lampiran 3. Dokumen ini dipilih sebagai sampel awal untuk menguji fungsionalitas program dalam menghasilkan *digital signature* menggunakan algoritma SHA-3 dan McEliece. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, hasil dari proses *hashing* menggunakan algoritma SHA-3 menghasilkan suatu *message digest* berupa deretan nilai heksadesimal. Hasil dari pengujian tersebut dapat dilihat pada Gambar 4.3.



Gambar 4.3 Hasil *Signing* Dokumen Asli dengan Kunci Pertama

Berdasarkan Gambar 4.3, diperoleh nilai *message digest* yang diperoleh dari proses *hashing* dengan algoritma SHA-3. Sebagaimana telah dijelaskan pada Kajian Teori, algoritma ini memiliki dua fase yaitu fase *absorbing* dan *squeezing* yang masing-masing pengujian akan dilakukan menggunakan bantuan program Python. Fase *absorbing* dilakukan dengan mengonversi dokumen dengan nama

dokumen1.pdf menggunakan UTF-8 dan dibagi menjadi blok-blok berukuran 136 byte (*rate*). Berdasarkan pengujian, diperoleh panjang pesan dokumen tersebut yaitu 986 byte dan setelah ditambahkan padding menjadi 1088 byte. Sehingga, pesan dari dokumen menghasilkan 8 blok berukuran 136 byte yang telah di XOR dari *state* S. Selanjutnya yaitu fase *squeezing*, di mana 136 byte pertama dari *state* terakhir diambil dan digunakan sebagai *output*. Berdasarkan pengujian diperoleh 8 *state*, sebagai berikut:

State awal =

$$0000000000000000000000000000000000000000000000000000000000000000$$

Setelah blok 1: *state* $[0:32] =$

$$9651313951979dcee9a3bfe1b29ff8b46c605ab25c769b$$

$$b5866fe640d8aca631$$

Setelah blok 2: *state* $[0:32] =$

$$3e80be856c07df3d36cdb64905043bfdf8322568e48493$$

$$8ed382637e18d84788$$

Setelah blok 3: *state* $[0:32] =$

$$c93c548fc3ab93c0e968260f6db554a20945ac68d808349$$

$$35e18a767d1dd6c88$$

Setelah blok 4: *state* $[0:32] =$

$$c8839d3991a802e4bb0c8cec92ef2bfdec8585f63f33c2$$

$$2f48b1120d9bdfbbf6$$

Setelah blok 5: *state* $[0:32] =$

$$06d0512d7c4d20471cd76d533d7b714dedf770ece099adf3$$

$$e0678b9dd8756415$$

Setelah blok 6: *state* $[0:32] =$

$$3c0a1254388194c1ce3e420cded3720c1adafc1036bae3$$

$$267fe1b3d1314ca8fe$$

Setelah blok 7: *state* $[0:32] =$

$$135fef5c2e19d2f0a59478c783f85bc9c9543fc2c7d6ad$$

$$b0cab4eaf29f174f5d$$

Setelah blok 8: *state* $[0:32] =$

$$657ee3871789ce1ca1eaaa782ec09eb08ad3d0e236e613$$

$$15f282e06b8a5a8e7b$$

Karena panjang *message digest* yang diinginkan adalah 256 bit (32 byte) dan ukuran tersebut lebih kecil dari 136 byte (*rate*), maka hanya diperlukan satu kali iterasi untuk menghasilkan *output* akhir, sebagai berikut:

$$657ee3871789ce1ca1eaaa782ec09eb08ad3d0e236e613$$

$$15f282e06b8a5a8e7b$$

Namun, perlu dingat bahwa pengujian fase *absorbing* dan *squeezing* ini hanyalah simulasi, hasil dapat berubah ketika program menjalankan fungsi Keccak-f[1600] sesuai spesifikasi FIPS-202. Berdasarkan pengujian sesuai dengan standar FIPS-202 menggunakan *library* sha3_256, diperoleh *message digest* sebagai berikut:

$$ac25ff13b2002ade1309c43777a2d439d3668090324b43$$

$$133707a1efbd97ed29$$

Message digest yang telah dihasilkan dari proses *hashing* tersebut dikonversi ke dalam bentuk bilangan biner. Proses konversi ini dilakukan sesuai dengan

penjelasan yang telah disampaikan pada Kajian Teori 2.7, yang membahas mengenai metode konversi bilangan dari bentuk heksadesimal ke bentuk biner. Melalui proses tersebut, diperoleh hasil konversi sebagai berikut:

1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0

0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 0

0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1

0 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0

0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0

0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 1

1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 1

Selanjutnya, *message digest* yang telah dikonversi ke dalam bentuk biner akan dibagi menjadi beberapa blok (P_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter (7,3), maka setiap blok akan dibagi menjadi 7 -bit. Proses pembagian dilakukan untuk mempermudah proses enkripsi dan memastikan setiap blok (P_i) dari *message digest* dapat diproses secara terpisah. Proses pembagian blok (P_i) menghasilkan blok-blok sebagai berikut:

$P_1 = [1\,0\,1\,0\,1\,1\,0]$ $P_8 = [0\,1\,0\,1\,0\,1\,0]$ $P_{15} = [1\,1\,0\,1\,1\,1\,1]$

$P_2 = [0\,0\,0\,1\,0\,0\,1]$ $P_9 = [1\,1\,0\,1\,1\,1\,1]$ $P_{16} = [0\,1\,0\,0\,0\,1\,0]$

$P_3 = [0\,1\,1\,1\,1\,1\,1]$ $P_{10} = [0\,0\,0\,0\,1\,0\,0]$ $P_{17} = [1\,1\,0\,1\,0\,1\,0]$

$P_4 = [1\,1\,1\,0\,0\,0\,1]$ $P_{11} = [1\,1\,0\,0\,0\,0\,1]$ $P_{18} = [0\,0\,0\,1\,1\,1\,0]$

$P_5 = [0\,0\,1\,1\,1\,0\,1]$ $P_{12} = [0\,0\,1\,1\,1\,0\,0]$ $P_{19} = [0\,1\,1\,1\,0\,1\,0]$

$P_6 = [1\,0\,0\,1\,0\,0\,0]$ $P_{13} = [0\,1\,0\,0\,0\,0\,1]$ $P_{20} = [0\,1\,1\,0\,1\,1\,0]$

$P_7 = [0\,0\,0\,0\,0\,0\,0]$ $P_{14} = [1\,0\,1\,1\,1\,0\,1]$ $P_{21} = [0\,1\,1\,0\,1\,0\,0]$

$P_{22} = [0\ 0\ 0\ 0\ 0\ 1\ 0]$ $P_{28} = [0\ 1\ 1\ 0\ 0\ 1\ 1]$ $P_{34} = [1\ 1\ 0\ 0\ 1\ 0\ 1]$

$P_{23} = [0\ 1\ 0\ 0\ 0\ 0\ 0]$ $P_{29} = [0\ 1\ 1\ 1\ 0\ 0\ 0]$ $P_{35} = [1\ 1\ 1\ 1\ 1\ 0\ 1]$

$P_{24} = [0\ 1\ 1\ 0\ 0\ 1\ 0]$ $P_{30} = [0\ 0\ 1\ 1\ 1\ 1\ 0]$ $P_{36} = [1\ 0\ 1\ 0\ 0\ 1\ 0]$

$P_{25} = [0\ 1\ 0\ 0\ 1\ 0\ 1]$ $P_{31} = [1\ 0\ 0\ 0\ 0\ 1\ 1]$ $P_{37} = [1\ 0\ 0\ 1]$

$P_{26} = [1\ 0\ 1\ 0\ 0\ 0\ 0]$ $P_{32} = [1\ 1\ 0\ 1\ 1\ 1\ 1]$

$P_{27} = [1\ 1\ 0\ 0\ 0\ 1\ 0]$ $P_{33} = [1\ 0\ 1\ 1\ 1\ 1\ 0]$

Selanjutnya, setiap blok *message digest* akan dienkripsi menggunakan kunci privat yang diperoleh melalui proses pembangkitan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.1, kunci privat yang diperoleh dengan parameter (7,3) yaitu

$$H' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (P_i) yang disebut sebagai vektor pesan (m), dengan matriks kunci privat (H') untuk memperoleh *ciphertext* (c). Selain itu, proses ini melibatkan vektor sindrom h berukuran ($n-k$)-bit, yaitu $h = [1\ 0\ 1\ 0]$ yang ditambahkan guna memperkuat proses enkripsi. Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c_{(n-k)-bit} = m_{n-bit} \cdot H'_{n\times(n-k)} + h_{(n-k)-bit}$$

$$c_{4-bit} = m_{7-bit} \cdot H'_{7\times4} + h_{4-bit}$$

$$c_1 = [1\ 0\ 1\ 0\ 1\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 0\ 0\ 1]$$

$$c_2 = [0\ 0\ 0\ 1\ 0\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 0\ 1]$$

$$c_3 = [0\ 1\ 1\ 1\ 1\ 1\ 1] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 1\ 0]$$

$$c_4 = [1\ 1\ 1\ 0\ 0\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 0\ 1]$$

$$c_5 = [0\ 0\ 1\ 1\ 1\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 0\ 1]$$

$$c_6 = [1\ 0\ 0\ 1\ 0\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 0]$$

$$c_7 = [0\ 0\ 0\ 0\ 0\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 1\ 0]$$

$$c_8 = [0\ 1\ 0\ 1\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 1\ 0]$$

$$c_9 = [1\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 0\ 0]$$

$$c_{10} = [0\ 0\ 0\ 0\ 1\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 1]$$

$$c_{11} = [1\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [0\ 0\ 1\ 0]$$

$$c_{12} = [0\ 0\ 1\ 1\ 1\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 0\ 1]$$

$$c_{13} = [0\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 1\ 1]$$

$$c_{14} = [1\ 0\ 1\ 1\ 1\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [0\ 0\ 0\ 0]$$

$$c_{15} = [1\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 0\ 0]$$

$$c_{16} = [0\ 1\ 0\ 0\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 0\ 1]$$

$$c_{17} = [1\ 1\ 0\ 1\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 0\ 1\ 1]$$

$$c_{18} = [0\ 0\ 0\ 1\ 1\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 0\ 0\ 0]$$

$$c_{19} = [0\ 1\ 1\ 1\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 1]$$

$$c_{20} = [0\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 0\ 1]$$

$$c_{21} = [0\ 1\ 1\ 0\ 1\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 1\ 1]$$

$$c_{22} = [0\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 0\ 0]$$

$$c_{23} = [0\ 1\ 0\ 0\ 0\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 1\ 1]$$

$$c_{24} = [0\ 1\ 1\ 0\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 0\ 1\ 0]$$

$$c_{25} = [0\ 1\ 0\ 0\ 1\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 0]$$

$$c_{26} = [1\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 0\ 0]$$

$$c_{27} = [1\ 1\ 0\ 0\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 0]$$

$$c_{28} = [0\ 1\ 1\ 0\ 0\ 1\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 1\ 0]$$

$$c_{29} = [0\ 1\ 1\ 1\ 0\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 1\ 1]$$

$$c_{30} = [0\ 0\ 1\ 1\ 1\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 1\ 1]$$

$$c_{31} = [1\ 0\ 0\ 0\ 0\ 1\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 1]$$

$$c_{32} = [1\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 0\ 0\ 0]$$

$$c_{33} = [1\ 0\ 1\ 1\ 1\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 1\ 0]$$

$$c_{34} = [1\ 1\ 0\ 0\ 1\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 0\ 1]$$

$$c_{35} = [1\ 1\ 1\ 1\ 1\ 0\ 1] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 1]$$

$$c_{36} = [1\ 0\ 1\ 0\ 0\ 1\ 0] \cdot H' + [1\ 0\ 1\ 0] = [0\ 1\ 1\ 0]$$

Karena $P_{37} = [1\ 0\ 0\ 1]$ tidak memenuhi ukuran 7-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $P_{37} = [1\ 0\ 0\ 1\ 0\ 0\ 0]$

$$c_{37} = [1\ 0\ 0\ 1\ 0\ 0\ 0] \cdot H' + [1\ 0\ 1\ 0] = [1\ 1\ 0\ 0]$$

Berdasarkan hasil enkripsi yang telah dilakukan, diperoleh *ciphertext* dari *dokumen1.pdf* dengan kunci pertama, sebagai berikut:

0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1

0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1

0 1 1 1 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 0

Ciphertext ini berfungsi sebagai *digital signature*, yang nantinya akan digunakan dalam proses verifikasi untuk memastikan keaslian dan integritas dokumen.

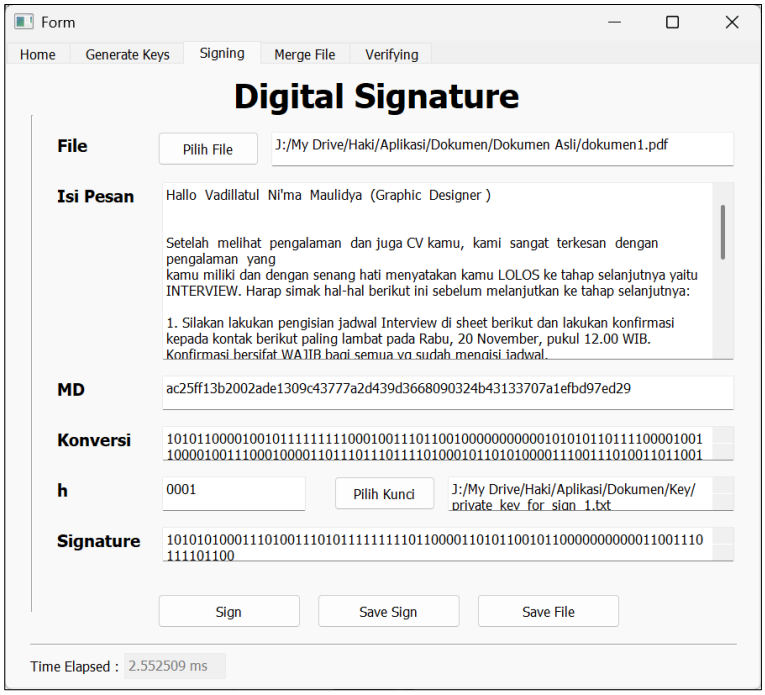
Selanjutnya, agar dapat dengan mudah dibaca oleh sistem verifikasi *digital signature* ini akan diubah menjadi QR Code berikut ini:

Gambar 4.4 QR Code Dokumen Asli dengan Kunci Pertama

Dokumen dan QR Code yang telah diperoleh akan digabungkan menjadi satu dokumen dan siap dikirimkan ke penerima.

4.2.2 Dokumen Asli dengan Kunci Kedua

Pengujian kedua dilakukan pada dokumen dengan konten yang sama seperti pada pengujian pertama, yaitu dokumen satu halaman dengan nama *dokumen1.pdf*, sebagaimana tercantum pada Lampiran 3. Perbedaan utama pada pengujian ini terletak pada penggunaan kunci yang berbeda untuk proses *signing* dan *verifying* dokumen. Tujuan dari pengujian ini untuk mengevaluasi pengaruh perubahan kunci terhadap hasil *digital signature*, serta memastikan bahwa sistem dapat mendeteksi perbedaan apabila terjadi ketidaksesuaian antara tanda tangan dan kunci yang digunakan. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, hasil dari proses *hashing* menggunakan algoritma SHA-3 menghasilkan suatu *message digest* berupa deretan nilai heksadesimal. Hasil dari pengujian tersebut dapat dilihat pada Gambar 4.5.



Gambar 4.5 Hasil *Signing* Dokumen Asli dengan Kunci Pertama

Berdasarkan Gambar 4.5, diperoleh nilai *message digest* yang diperoleh dari proses *hashing* dengan algoritma SHA-3. Sebagaimana telah dijelaskan pada sub-bab sebelumnya cara memperoleh *message digest* dari suatu dokumen, sehingga pada dokumen tersebut diperoleh *message digest* sebagai berikut:

$$ac25ff13b2002ade1309c43777a2d439d3668090324b43$$

$$133707a1efbd97ed29$$

Selanjutnya, *message digest* yang telah dihasilkan dari proses *hashing* tersebut dikonversi ke dalam bentuk bilangan biner. Proses konversi ini dilakukan sesuai dengan penjelasan yang telah disampaikan pada Kajian Teori 2.7, yang membahas mengenai metode konversi bilangan dari bentuk heksadesimal ke bentuk biner. Melalui proses tersebut, diperoleh hasil konversi sebagai berikut:

$$1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0\,1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0$$

$$0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0$$

$$0\,0\,1\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,0\,0\,1$$

0 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0

0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0

0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 1

1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 1

Selanjutnya, *messege digest* yang telah dikonversi ke dalam bentuk biner akan dibagi menjadi beberapa blok (P_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter (13,9), maka setiap blok akan dibagi menjadi 13 -bit. Proses pembagian dilakukan untuk mempermudah proses enkripsi dan memastikan setiap blok (P_i) dari *messege digest* dapat diproses secara terpisah. Proses pembagian blok (P_i) menghasilkan blok-blok sebagai berikut:

$P_1 = [1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0]$

$P_2 = [1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0]$

$P_3 = [0\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,1]$

$P_4 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0]$

$P_5 = [1\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0]$

$P_6 = [0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0\,1\,0]$

$P_7 = [0\,1\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1]$

$P_8 = [1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1]$

$P_9 = [1\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1\,0]$

$P_{10} = [1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1]$

$P_{11} = [0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,1]$

$P_{12} = [0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1]$

$P_{13} = [0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0]$

$P_{14} = [1\,0\,0\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0]$

$P_{15} = [1\,1\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1]$

$P_{16} = [1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1]$

$P_{17} = [1\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1]$

$P_{18} = [1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0]$

$P_{19} = [0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,1\,1\,0]$

$P_{20} = [1\,0\,0\,1\,0\,1\,0\,0\,1]$

Selanjutnya, setiap blok *message digest* akan dienkripsi menggunakan kunci privat yang diperoleh melalui proses pembangkitan kunci yang mengacu pada

parameter yang telah ditentukan. Berdasarkan Gambar 4.2, kunci privat yang diperoleh dengan parameter $(13,9)$ yaitu

$$H' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (P_i) yang disebut sebagai vektor pesan (m), dengan matriks kunci privat (H') untuk memperoleh *ciphertext* (c). Selain itu, proses ini melibatkan vektor sindrom h berukuran $(n-k)$-bit, yaitu $h = [0\ 0\ 0\ 1]$ yang ditambahkan guna memperkuat proses enkripsi. Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c_{(n-k)-bit} = m_{n-bit} \cdot H'_{n\times(n-k)} + h_{(n-k)-bit}$$

$$c_{4-bit} = m_{13-bit} \cdot H'_{13\times4} + h_{4-bit}$$

$$c_1 = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 0\ 1\ 0]$$

$$c_2 = [1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 0\ 1\ 0]$$

$$c_3 = [0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1] \cdot H' + [0\ 0\ 0\ 1] = [0\ 0\ 1\ 1]$$

$$c_4 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 0\ 1\ 0]$$

$$c_5 = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0] \cdot H' + [0\ 0\ 0\ 1] = [0\ 1\ 1\ 1]$$

$$c_6 = [0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H' + [0\ 0\ 0\ 1] = [0\ 1\ 0\ 1]$$

$$c_7 = [0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H' + [0\ 0\ 0\ 1] = [1\ 1\ 1\ 1]$$

$$c_8 = [1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1] \cdot H' + [0\ 0\ 0\ 1] = [1\ 1\ 1\ 1]$$

$$c_9 = [1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0] \cdot H' + [0\ 0\ 0\ 1] = [0\ 1\ 1\ 0]$$

$$c_{10} = [1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1] \cdot H' + [0\ 0\ 0\ 1] = [0\ 0\ 0\ 1]$$

$$c_{11} = [0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1] \cdot H' + [0\ 0\ 0\ 1] = [1\ 0\ 1\ 0]$$

$$c_{12} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1] \cdot H' + [0\ 0\ 0\ 1] = [1\ 1\ 0\ 0]$$

$$c_{13} = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 0\ 1\ 1]$$

$$c_{14} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H' + [0\ 0\ 0\ 1] = [0\ 0\ 0\ 0]$$

$$c_{15} = [1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1] \cdot H' + [0\ 0\ 0\ 1] = [0\ 0\ 0\ 0]$$

$$c_{16} = [1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1] \cdot H' + [0\ 0\ 0\ 1] = [0\ 0\ 0\ 1]$$

$$c_{17} = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1] \cdot H' + [0\ 0\ 0\ 1] = [1\ 0\ 0\ 1]$$

$$c_{18} = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 1\ 0\ 1]$$

$$c_{19} = [0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 1\ 1\ 0]$$

Karena $P_{20} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1]$ tidak memenuhi ukuran 13-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $P_{20} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$

$$c_{20} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H' + [0\ 0\ 0\ 1] = [1\ 1\ 0\ 0]$$

Berdasarkan hasil enkripsi yang telah dilakukan, diperoleh *ciphertext* dari *dokumen1.pdf* dengan kunci kedua, sebagai berikut:

1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 0

0 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1

1 0 1 1 0 0

Ciphertext ini berfungsi sebagai *digital signature*, yang nantinya akan digunakan dalam proses verifikasi untuk memastikan keaslian dan integritas dokumen.

Selanjutnya, agar dapat dengan mudah dibaca oleh sistem verifikasi *digital signature* ini akan diubah menjadi QR Code menjadi

Gambar 4.6 QR Code Dokumen Asli dengan Kunci Kedua

Dokumen dan QR Code yang telah diperoleh akan digabungkan menjadi satu dokumen dan dikirimkan ke penerima.

Berdasarkan hasil dari kedua pengujian yang telah dilakukan dengan menggunakan dokumen yang sama, diperoleh bahwa perbedaan kunci yang digunakan dalam proses *signing* menghasilkan *digital signature* yang berbeda. Hasil dari kedua pengujian tersebut dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil *Signing* Menggunakan Satu Dokumen yang Sama

Massage Digest	**Parameter**	***Digital Signature***
$ac25ff13b2002ad$ $e1309c43777a2$ $d439d3668090$ $324b43133707a1$ $efbd97ed29$	(7,3)	10101010001110100111010111111111011000011010110010110000000000011001110111101100
	(13,9)	0001100101100101100111001010101010001101001001011011000010000101001100001101010101111000011100101100010011001110111101111011000111001011101011011100

Hasil pengujian terhadap dokumen yang menggunakan dua kunci berbeda menunjukkan bahwa program memiliki kemampuan yang baik dalam membedakan hasil enkripsi berdasarkan variasi kunci yang diterapkan. Perbedaan ini terlihat

secara jelas karena algoritma McEliece yang dapat mempengaruhi setiap *digital signature* yang dihasilkan. Di mana setiap pasangan kunci menghasilkan *output* yang berbeda meskipun proses *signing* dilakukan terhadap dokumen yang sama. Hal ini menunjukkan bahwa sistem berhasil menjaga sensitivitas terhadap perubahan kunci, yang merupakan salah satu indikator penting dalam mekanisme keamanan *digital signature*. Selain itu, program juga terbukti mampu menjaga keunikan *digital signature* yang dihasilkan, serta memastikan bahwa setiap dokumen yang ditandatangani memiliki identitas yang berbeda meskipun dokumen memiliki konten yang sama.

Adapun hasil pengujian pada proses *signing* dokumen dari dua puluh dokumen yang dapat dilihat pada Tabel 4.2. Tabel tersebut menyajikan pasangan kunci privat yang digunakan serta digital signature yang dihasilkan dari proses enkripsi dokumen. *Digital signature* yang dihasilkan kemudian dikombinasikan dengan isi dokumen dan disiapkan untuk dikirimkan kepada pihak penerima. Penggabungan antara dokumen dan *digital signature* memungkinkan penerima melakukan proses verifikasi menggunakan kunci publik yang sesuai agar dapat memastikan keaslian dokumen yang diterima. Dengan demikian, sistem tidak hanya menunjukkan keberhasilan dalam aspek teknis enkripsi dan penandatanganan, tetapi juga dalam mendukung prinsip keaslian, integritas, dan *non-repudiation* dalam komunikasi digital yang aman.

Tabel 4.2 Hasil *Signing* dari Dua Puluh Dokumen

No.	Nama Dokumen	*Message Digest*	Kunci Privat	*Digital Signature*
1	dokumen1	$ac25ff13b2002ade1$ $309c43777a2d439d$ $3668090324b4313$ $3707a1efbd97ed29$	$H' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$ dan $h = [1\ 0\ 1\ 0]$	10101010001110100111010111111111101100 00110101100101100000000000011001110111 101100
	dokumen1	$ac25ff13b2002ade1$ $309c43777a2d439d$ $3668090324b431337$ $07a1efbd97ed29$	$H' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$ dan $h = [0\ 0\ 0\ 1]$	00011001011001011001110010101010001 10100100101101100001000010100110000011 01010101111000011100101100010011001111 01111011111011000111001011101011011100
2	dokumen2	$644d976b9605dd1ed$ $728a330057621dbe91$ $3851edb06c1840f79$ $dd5ced5b0cd0$	$H' = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$	00100011001011001100101111011000011001 01101010111111000110111111001101110011 01000101011010010001010100000000000001 11000011111011100110000010010011110 10

No.	Nama Dokumen	*Message Digest*	Kunci Privat	*Digital Signature*
3	dokumen3	$1dc88e9102fa5985$ $911a0121f6a4f7f79$ $082944c9962795d95$ $19e6a482109851$	dan $h = [1\ 0\ 1\ 0]$	00001011000111010101101111100111000 11 00110010111100100111010000010001101 10 11010110101000111011101100011101100 11 11000011101110000111111010011000101 01
4	dokumen4	$a363a1c2d484e24$ $dd53940013bbb67c$ $af4cb119f3baa633$ $cd18d1be6f073de9c$		10001011111001011000011010011100001 00 01011101010101011010000000111101011 11 10101110110110101000011111001001110 11 10010011110011011011001011000110111 10
5	dokumen5	$274912c92e6ab6559$ $bb5c5b95c91c6bced$ $5bfa03d0ce2aed3d$ $5c469c367728f2$	$H' = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ dan $h = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1]$	10011110001000011010000001011101001 11 10010010101000101010011100001101001 01 01010011001100110100010001000110011 11 00100011111110111
6	dokumen6	$fb938bcd80ac28a15$ $de8824779f8b5fff6$ $3b96be9c0033e5cd0$ $1425ec6753638$		11001110100110111011110010001110001 10 00010111101011010010110011100010101 11 00000100001000110011100011010101101 01 00011110111110111
7	dokumen7	$ff494ea8eedec53d$ $77021cb3d3617dc2$ $e00a5737ba8c7de$ $36327a568b9531a53$		01100011011010110001100101000011111 00 00010110101100001100111000101100011 00 10001000111110001001111100010010011 11 00001010000101010
8	dokumen8	$34032b03846e3f$ $1e9a117d7757c2a4$ $4b1479bb76864a4f$ $12f6517f42af3bdcb5$		11000110110001111010011110010000110 11 10111111011100010101110010110011001 00 01100001111000111100011010001000100 01 11100101100101010

No.	Nama Dokumen	*Message Digest*	Kunci Privat	*Digital Signature*
9	dokumen9	$ac4ce85df44b1f$ $c7f322e423f06fb4$ $2853acab1cacfcba$ $61fac6b6837287bb15$	$H' = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ dan $h = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1]$	11111111010000011000100101101111011100011100110011101101011110010111000100100010100111111011100010101000111101110100000110011
10	dokumen10	$f125d9d5b8d85c786$ $06e1d6d3ae3e08d5$ $d2c5e747081e0b60$ $594eb246f0c8de3$		00110010001100001100111110101011011010111100101001101111101100001100111011010011000010000100001100111110010001100
11	dokumen11	$9d19420327742912$ $692a92878c80c91b$ $a5d3bda30e13dc6d$ $05bbe0933cec6d1e$		10110111010110010001111010101111111011010010100011101001111101100001011101100010001110110101100110011101111010100111011100
12	dokumen12	$51cd164131ad476c$ $df7cf8d119a3a838$ $c4edf6651763945a29$ $bff15c8f853b59$		00011111000110101101101010110001010100010101110111001111001101101100000001010110100111010100000011101010000000010011001010001100
13	dokumen13	$5848bd7342954326e$ $990493fe2da3f0ce$ $1bcb785db4cbe6fed$ $1361194fb51c2b$		01011000010011110101000011111101110101001111000110100001011111010110100111111111011000000101
14	dokumen14	$0c4a13abece3294$ $f8b3c585e594808d$ $ea0aabd3abe6c442$ $37e3b46828b421009$		01101111101011010100111001101111110100111101110010010101100111011010011111011110001111111000101

No.	Nama Dokumen	*Message Digest*	Kunci Privat	*Digital Signature*
15	dokumen15	$ff686e4246483100e$ $43d207b2c7841c42$ $d37dfc45b369a90$ $23e0bab7405fa5c3$	$H' = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ dan $h = [1\ 1\ 0\ 1\ 1\ 0]$	00001001110000101100111001110000011100100000011110000111011111101101011010001001111011011101000101
16	dokumen16	$308232872f4ace00$ $a63897d0d86cfd4a$ $5f356ea867e8378$ $caa4690a3d7fa5a12$		00100111100000011000110110010110100010000011010111101101011110001011101001110010010010000011010
17	dokumen17	$9f75d6410488d57ce$ $e015f1cbc7638b580$ $d7c841ebd3e7c1c3fd$ $8cd31eb09b9d$		10011000001000101110010000101011111101011000100
18	dokumen18	$ef1c00dbcc3d953$ $3223cb1a8ad3aa7$ $2b4268fc35bca062$ $4a70ebfa5eadab3733$		00001100110011001101101001100000100010011101011

No.	Nama Dokumen	*Message Digest*	Kunci Privat	*Digital Signature*
19	dokumen19	$cb5928bde1ca85a$ $1fa24907b294a092$ $4b7c2419578e16a5$ $c7bfe36323d045e1f$	$H' = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$ dan $h = [1\ 0\ 1\ 0]$	00001010110001011111111000100000000000000100111
20	dokumen20	$6451543643677d91f$ $48aadf1d98584630e$ $c0439c09380ece597$ $771497d1fa4a2$		01011000001111101100100000101101010010000111001

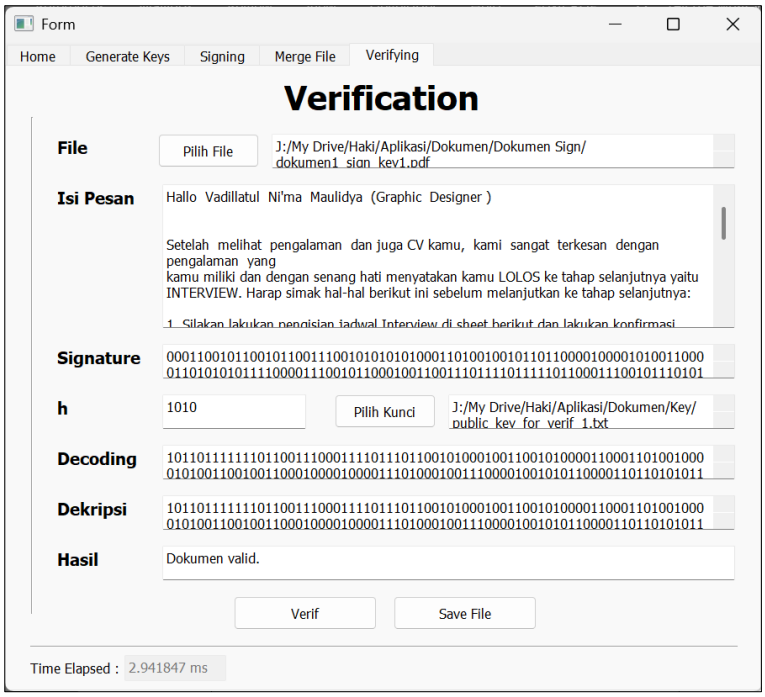
4.3 *Verifying* Dokumen

Proses verifikasi dokumen terdiri dari dua langkah utama yang saling berkaitan. Langkah pertama adalah proses dekripsi *ciphertext* berupa *digital signature* yang disimpan dalam bentuk QR Code pada dokumen yang diterima. QR Code ini dipindai dan diproses untuk memperoleh kembali *message digest.* Langkah kedua adalah proses *decoding* isi dokumen yang diterima, yaitu dengan menghitung nilai *hash* dari isi dokumen menggunakan algoritma SHA-3 dan mengonversinya ke dalam bentuk biner. Kedua hasil tersebut akan dibandingkan dan jika nilai yang diperoleh dari hasil kedua langkah tersebut sama, maka dokumen yang diterima dianggap valid atau tidak mengalami perubahan selama proses pengiriman. Sebaliknya, jika terdapat perubahan pada isi dokumen, maka sistem akan menghasilkan nilai yang berbeda dan dokumen dianggap tidak valid.

Metode ini diterapkan pada dua puluh dokumen berformat .pdf, berisi alfabet, angka, dan karakter. Seluruh proses dilakukan menggunakan bahasa pemrograman Python untuk menjamin efisiensi, fleksibilitas, dan integrasi dengan berbagai pustaka kriptografi modern. Untuk mendukung kenyamanan dan kemudahan penggunaan, tampilan antarmuka sistem dirancang menggunakan PyQt5. Hal ini bermaksud memungkinkan pengguna berinteraksi dengan sistem secara visual, mulai dari pemilihan dokumen, pemindaian *digital signature* dalam bentuk QR Code, hingga proses *hashing* dan verifikasi dokumen. Dengan demikian, sistem tidak hanya mendukung aspek keamanan data, tetapi juga memperhatikan aspek *user interface* dalam proses otentikasi dokumen digital.

4.3.1 Dokumen Asli dengan Kunci Pertama

Verifikasi pertama dilakukan pada dokumen satu halaman dengan nama *dokumen1_sign_key1.pdf*, sebagaimana tercantum pada Lampiran 4. Dokumen ini telah diterima sebagai sampel awal untuk menguji fungsionalitas program dalam memverifikasi dokumen menggunakan algoritma SHA-3 dan McEliece. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.7.



Gambar 4.7 Hasil *Verifying* Dokumen Asli dengan Kunci Pertama

Berdasarkan Gambar 4.7, dapat dilihat bahwa hasil dari pemindaian QR Code menghasilkan *digital signature* berupa biner yang sesuai dengan hasil enkripsi pada proses *signing* sebelumnya. Nilai *signature* yang diperoleh dari hasil pemindaian QR Code adalah

0 0 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 1 0 0 0 1

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1

0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1

0 1 1 1 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 0

Selanjutnya, *signature* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran $(n-k)$ -bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter (7,3), maka setiap blok akan dibagi menjadi 4 -bit. Proses pembagian ini dilakukan untuk mempermudah proses dekripsi memastikan setiap blok (C_i) dari *digital signature* dapat diproses secara terpisah. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [0\ 0\ 0\ 1]$ $C_{14} = [0\ 0\ 0\ 0]$ $C_{27} = [1\ 1\ 0\ 0]$

$C_2 = [1\ 0\ 0\ 1]$ $C_{15} = [1\ 0\ 0\ 0]$ $C_{28} = [1\ 1\ 1\ 0]$

$C_3 = [0\ 1\ 1\ 0]$ $C_{16} = [0\ 1\ 0\ 1]$ $C_{29} = [1\ 1\ 1\ 1]$

$C_4 = [0\ 1\ 0\ 1]$ $C_{17} = [0\ 0\ 1\ 1]$ $C_{30} = [0\ 1\ 1\ 1]$

$C_5 = [1\ 0\ 0\ 1]$ $C_{18} = [0\ 0\ 0\ 0]$ $C_{31} = [1\ 1\ 0\ 1]$

$C_6 = [1\ 1\ 0\ 0]$ $C_{19} = [1\ 1\ 0\ 1]$ $C_{32} = [1\ 0\ 0\ 0]$

$C_7 = [1\ 0\ 1\ 0]$ $C_{20} = [0\ 1\ 0\ 1]$ $C_{33} = [1\ 1\ 1\ 0]$

$C_8 = [1\ 0\ 1\ 0]$ $C_{21} = [0\ 1\ 1\ 1]$ $C_{34} = [0\ 1\ 0\ 1]$

$C_9 = [1\ 0\ 0\ 0]$ $C_{22} = [1\ 0\ 0\ 0]$ $C_{35} = [1\ 1\ 0\ 1]$

$C_{10} = [1\ 1\ 0\ 1]$ $C_{23} = [0\ 1\ 1\ 1]$ $C_{36} = [0\ 1\ 1\ 0]$

$C_{11} = [0\ 0\ 1\ 0]$ $C_{24} = [0\ 0\ 1\ 0]$ $C_{37} = [1\ 1\ 0\ 0]$

$C_{12} = [0\ 1\ 0\ 1]$ $C_{25} = [1\ 1\ 0\ 0]$

$C_{13} = [1\ 0\ 1\ 1]$ $C_{26} = [0\ 1\ 0\ 0]$

Selanjutnya, setiap blok *ciphertext* akan didekripsi menggunakan kunci publik yang diperoleh melalui proses membangkitkan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.1, kunci publik yang diperoleh dengan parameter (7,3) yaitu:

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang disebut sebagai *ciphertext* (c), dengan invers dari matriks pengacak (S)

$$S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = c_{4-bit} \cdot S^{-1}_{4\times4}$$

$c'_1 = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{11} = [0\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 0]$

$c'_2 = [1\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 1]$ $\quad$ $c'_{12} = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_3 = [0\ 1\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 1]$ $\quad$ $c'_{13} = [1\ 0\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$

$c'_4 = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$ $\quad$ $c'_{14} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_5 = [1\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 1]$ $\quad$ $c'_{15} = [1\ 0\ 0\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$

$c'_6 = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$ $\quad$ $c'_{16} = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_7 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 0]$ $\quad$ $c'_{17} = [0\ 0\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$

$c'_8 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 0]$ $\quad$ $c'_{18} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_9 = [1\ 0\ 0\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$ $\quad$ $c'_{19} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$

$c'_{10} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$ $\quad$ $c'_{20} = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_{21} = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$

$c'_{22} = [1\ 0\ 0\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$

$c'_{23} = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$

$c'_{24} = [0\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 0]$

$c'_{25} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

$c'_{26} = [0\ 1\ 0\ 0] \cdot S^{-1} = [1\ 1\ 0\ 1]$

$c'_{27} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

$c'_{28} = [1\ 1\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 1]$

$c'_{29} = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$

$c'_{30} = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$

$c'_{31} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$

$c'_{32} = [1\ 0\ 0\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$

$c'_{33} = [1\ 1\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 1]$

$c'_{34} = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_{35} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$

$c'_{36} = [0\ 1\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 1]$

$c'_{37} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

Berdasarkan hasil dekripsi yang telah dilakukan, diperoleh dekripsi *ciphertext* dari *dokumen1_sign_key1.pdf* dengan kunci pertama, sebagai berikut:

1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1

0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0

1 0 0 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1

1 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0 1

Selanjutnya, langkah kedua dalam proses verifikasi yaitu melakukan *decoding* dari isi dokumen *dokumen1_sign_key1.pdf* yang diterima. Proses *decoding* ini bertujuan untuk mengekstrak informasi asli dari dokumen dan membandingkannya dengan hasil dekripsi *digital signature*. Hasil *decoding* isi dokumen yang diterima diperoleh, sebagai berikut:

1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0

0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 0

0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1

0 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0

0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0

0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 1

1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 1

Selanjutnya, hasil *decoding* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(7,3)$, maka setiap blok akan dibagi menjadi 7-bit. Proses pembagian ini dilakukan untuk mempermudah proses *decoding* dari dokumen. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\ 0\ 1\ 0\ 1\ 1\ 0]$

$C_2 = [0\ 0\ 0\ 1\ 0\ 0\ 1]$

$C_3 = [0\ 1\ 1\ 1\ 1\ 1\ 1]$

$C_4 = [1\ 1\ 1\ 0\ 0\ 0\ 1]$

$C_5 = [0\ 0\ 1\ 1\ 1\ 0\ 1]$

$C_6 = [1\ 0\ 0\ 1\ 0\ 0\ 0]$

$C_7 = [0\ 0\ 0\ 0\ 0\ 0\ 0]$

$C_8 = [0\ 1\ 0\ 1\ 0\ 1\ 0]$

$C_9 = [1\ 1\ 0\ 1\ 1\ 1\ 1]$

$C_{10} = [0\ 0\ 0\ 0\ 1\ 0\ 0]$

$C_{11} = [1\ 1\ 0\ 0\ 0\ 0\ 1]$

$C_{12} = [0\ 0\ 1\ 1\ 1\ 0\ 0]$

$C_{13} = [0\ 1\ 0\ 0\ 0\ 0\ 1]$

$C_{14} = [1\ 0\ 1\ 1\ 1\ 0\ 1]$

$C_{15} = [1\ 1\ 0\ 1\ 1\ 1\ 1]$

$C_{16} = [0\ 1\ 0\ 0\ 0\ 1\ 0]$

$C_{17} = [1\ 1\ 0\ 1\ 0\ 1\ 0]$

$C_{18} = [0\ 0\ 0\ 1\ 1\ 1\ 0]$

$C_{19} = [0\ 1\ 1\ 1\ 0\ 1\ 0]$

$C_{20} = [0\ 1\ 1\ 0\ 1\ 1\ 0]$

$C_{21} = [0\ 1\ 1\ 0\ 1\ 0\ 0]$

$C_{22} = [0\ 0\ 0\ 0\ 0\ 1\ 0]$

$C_{23} = [0\ 1\ 0\ 0\ 0\ 0\ 0]$

$C_{24} = [0\ 1\ 1\ 0\ 0\ 1\ 0]$

$C_{25} = [0\ 1\ 0\ 0\ 1\ 0\ 1]$

$C_{26} = [1\ 0\ 1\ 0\ 0\ 0\ 0]$

$C_{27} = [1\ 1\ 0\ 0\ 0\ 1\ 0]$

$C_{28} = [0\ 1\ 1\ 0\ 0\ 1\ 1]$

$C_{29} = [0\ 1\ 1\ 1\ 0\ 0\ 0]$

$C_{30} = [0\ 0\ 1\ 1\ 1\ 1\ 0]$

$C_{31} = [1\ 0\ 0\ 0\ 0\ 1\ 1]$

$C_{32} = [1\ 1\ 0\ 1\ 1\ 1\ 1]$

$C_{33} = [1\ 0\ 1\ 1\ 1\ 1\ 0]$

$C_{34} = [1\ 1\ 0\ 0\ 1\ 0\ 1]$

$C_{35} = [1\ 1\ 1\ 1\ 1\ 0\ 1]$

$C_{36} = [1\ 0\ 1\ 0\ 0\ 1\ 0]$

$C_{37} = [1\ 0\ 0\ 1]$

Selanjutnya, setiap blok *decoding* akan dilakukan operasi perkalian menggunakan kunci publik untuk dibandingkan dengan hasil dekripsi *ciphertext* yang telah dihitung sebelumnya. Kunci publik diperoleh melalui proses membangkitkan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.1, kunci publik yang diperoleh dengan parameter $(7,3)$ yaitu

$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ dan } S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai pesan (m), dengan transpose matriks *parity-check* (H^T) ditambah dengan invers dari matriks pengacak (S)

$$H^T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \text{ dan } S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Selain itu, proses ini melibatkan vektor sindrom h berukuran $(n-k)$-bit, yaitu $h = [1\ 0\ 1\ 0]$ yang ditambahkan untuk memulihkan *noise* yang diberikan. Hasil perhitungan tersebut sebagai berikut:

$$c'_{(n-k)-bit} = m_{n-bit} H^T_{n\times(n-k)} + h_{(n-k)-bit} S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = m_{7-bit} H^T_{7\times4} + h_{4-bit} S^{-1}_{4\times4}$$

$$c'_1 = [1\ 0\ 1\ 0\ 1\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c'_2 = [0\ 0\ 0\ 1\ 0\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 1]$$

$$c'_3 = [0\ 1\ 1\ 1\ 1\ 1\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

$$c'_4 = [1\ 1\ 1\ 0\ 0\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_5 = [0\ 0\ 1\ 1\ 1\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 1]$$

$$c'_6 = [1\ 0\ 0\ 1\ 0\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

$$c'_7 = [0\ 0\ 0\ 0\ 0\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 0]$$

$$c'_8 = [0\ 1\ 0\ 1\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 0]$$

$$c'_9 = [1\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$$

$$c'_{10} = [0\ 0\ 0\ 0\ 1\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_{11} = [1\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{12} = [0\ 0\ 1\ 1\ 1\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{13} = [0\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 0\ 1]$$

$$c'_{14} = [1\ 0\ 1\ 1\ 1\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c'_{15} = [1\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$$

$$c'_{16} = [0\ 1\ 0\ 0\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{17} = [1\ 1\ 0\ 1\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c'_{18} = [0\ 0\ 0\ 1\ 1\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c'_{19} = [0\ 1\ 1\ 1\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_{20} = [0\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{21} = [0\ 1\ 1\ 0\ 1\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c'_{22} = [0\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$$

$$c'_{23} = [0\ 1\ 0\ 0\ 0\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c'_{24} = [0\ 1\ 1\ 0\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{25} = [0\ 1\ 0\ 0\ 1\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

$$c'_{26} = [1\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 1]$$

$$c'_{27} = [1\ 1\ 0\ 0\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

$$c'_{28} = [0\ 1\ 1\ 0\ 0\ 1\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 1]$$

$$c'_{29} = [0\ 1\ 1\ 1\ 0\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 0\ 0]$$

$$c'_{30} = [0\ 0\ 1\ 1\ 1\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c'_{31} = [1\ 0\ 0\ 0\ 0\ 1\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_{32} = [1\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 0]$$

$$c'_{33} = [1\ 0\ 1\ 1\ 1\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 1\ 1]$$

$$c'_{34} = [1\ 1\ 0\ 0\ 1\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{35} = [1\ 1\ 1\ 1\ 1\ 0\ 1] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_{36} = [1\ 0\ 1\ 0\ 0\ 1\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

Karena $c'_{37} = [1\ 0\ 0\ 1]$ tidak memenuhi ukuran 7-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $c'_{37} = [1\ 0\ 0\ 1\ 0\ 0\ 0]$

$$c'_{37} = [1\ 0\ 0\ 1\ 0\ 0\ 0] \cdot H^T + [1\ 0\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

Berdasarkan hasil *decoding* yang telah dilakukan, diperoleh *decoding* isi dokumen dari *dokumen1_sign_key1.pdf* dengan kunci pertama, sebagai berikut:

1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1

0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0

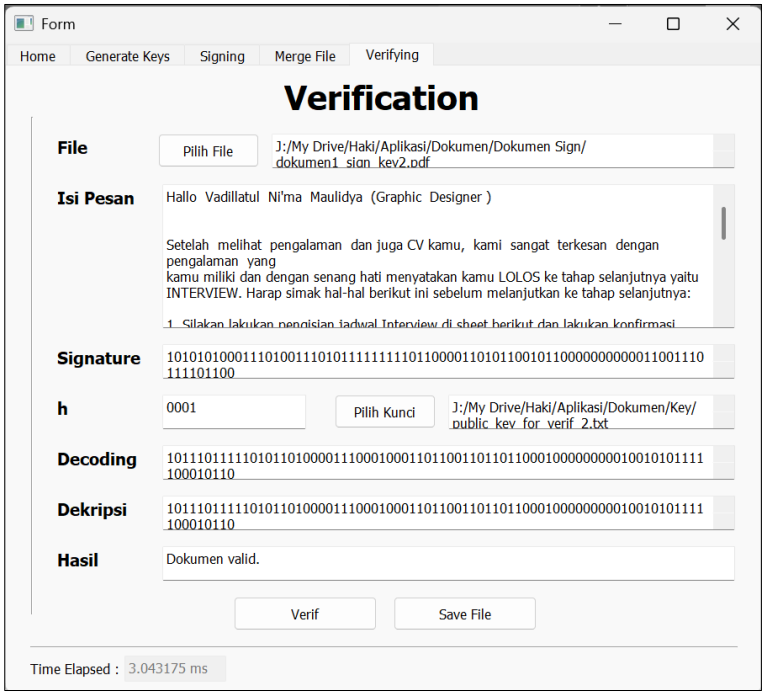
1 0 0 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1

1 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0 1

Berdasarkan perhitungan kedua langkah yang telah dilakukan, diperoleh hasil dekripsi *ciphertext* dan *decoding* isi dokumen menghasilkan nilai yang sama. Dengan demikian, dokumen dengan nama *dokumen1_sign_key1.pdf* yang diterima dianggap valid, hal ini menandakan bahwa dokumen tidak mengalami perubahan selama proses pengiriman.

4.3.2 Dokumen Asli dengan Kunci Kedua

Pengujian kedua dilakukan pada dokumen dengan konten yang sama seperti pada pengujian pertama, yaitu dokumen berjumlah satu halaman dengan nama *dokumen1_sign_key2.pdf*, sebagaimana tercantum dalam Lampiran 5. Perbedaan utama pada pengujian ini terletak pada penggunaan kunci yang berbeda untuk proses *signing* dan *verifying* digital. Tujuan dari pengujian ini untuk mengevaluasi pengaruh perubahan kunci terhadap hasil *digital signature*, serta memastikan bahwa sistem dapat mendeteksi perbedaan apabila terjadi ketidaksesuaian antara tanda tangan dan kunci yang digunakan. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.8.



Gambar 4.8 Hasil *Verifying* Dokumen Asli dengan Kunci Kedua

Berdasarkan Gambar 4.8, dapat dilihat bahwa hasil dari pemindaian QR Code menghasilkan *digital signature* berupa biner yang sesuai dengan hasil enkripsi pada proses *signing* sebelumnya. Nilai *signature* yang diperoleh dari hasil pemindaian QR Code adalah

$$1\,0\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,1\,0\,1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0$$

$$0\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,1\,1\,0\,1\,1\,1$$

$$1\,0\,1\,1\,0\,0$$

Selanjutnya, *signature* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran $(n-k)$ -bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(13,9)$, maka setiap blok akan dibagi menjadi 4 -bit. Proses pembagian ini dilakukan untuk mempermudah proses dekripsi memastikan setiap blok (C_i) dari *digital signature* dapat diproses

secara terpisah. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\ 0\ 1\ 0]$ $C_8 = [1\ 1\ 1\ 1]$ $C_{15} = [0\ 0\ 0\ 0]$

$C_2 = [1\ 0\ 1\ 0]$ $C_9 = [0\ 1\ 1\ 0]$ $C_{16} = [0\ 0\ 0\ 1]$

$C_3 = [0\ 0\ 1\ 1]$ $C_{10} = [0\ 0\ 0\ 1]$ $C_{17} = [1\ 0\ 0\ 1]$

$C_4 = [1\ 0\ 1\ 0]$ $C_{11} = [1\ 0\ 1\ 0]$ $C_{18} = [1\ 1\ 0\ 1]$

$C_5 = [0\ 1\ 1\ 1]$ $C_{12} = [1\ 1\ 0\ 0]$ $C_{19} = [1\ 1\ 1\ 0]$

$C_6 = [0\ 1\ 0\ 1]$ $C_{13} = [1\ 0\ 1\ 1]$ $C_{20} = [1\ 1\ 0\ 0]$

$C_7 = [1\ 1\ 1\ 1]$ $C_{14} = [0\ 0\ 0\ 0]$

Selanjutnya, setiap blok *ciphertext* akan didekripsi menggunakan kunci publik yang diperoleh melalui proses membangkitkan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter (13,9) yaitu

$$S = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang disebut sebagai *ciphertext* (c), dengan invers dari matriks pengacak (S)

$$S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = c_{4-bit} \cdot S^{-1}_{4\times4}$$

$c'_1 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $c'_2 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$

$c'_3 = [0\ 0\ 1\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$ $\quad$ $c'_{12} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_4 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{13} = [1\ 0\ 1\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$

$c'_5 = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$ $\quad$ $c'_{14} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_6 = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$ $\quad$ $c'_{15} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_7 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $\quad$ $c'_{16} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$

$c'_8 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $\quad$ $c'_{17} = [1\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$

$c'_9 = [0\ 1\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 1]$ $\quad$ $c'_{18} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$

$c'_{10} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$ $\quad$ $c'_{19} = [1\ 1\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

$c'_{11} = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{20} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

Berdasarkan hasil dekripsi yang telah dilakukan, diperoleh dekripsi *ciphertext* dari *dokumen_sign_key2.pdf* dengan kunci kedua, sebagai berikut:

1 0 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1

0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0

0 1 0 1 1 0

Selanjutnya, langkah kedua dalam proses verifikasi yaitu melakukan *decoding* terhadap isi dokumen *dokumen1_sign_key2.pdf* yang diterima. Proses *decoding* ini bertujuan untuk mengekstrak informasi asli dari dokumen dan membandingkannya dengan hasil dekripsi *digital signature*. Hasil *decoding* isi dokumen yang diterima diperoleh, sebagai berikut:

1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 1 1 0

0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 0

0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1

0 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 0 0

0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0

0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 1

1 1 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 0 0 1

Selanjutnya, hasil *decoding* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(13,9)$, maka setiap blok akan dibagi menjadi 13-bit. Proses pembagian ini dilakukan untuk mempermudah proses *decoding* dari dokumen. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\,0\,1\,0\,1\,1\,0\,0\,0\,0\,1\,0\,0]$ $C_{11} = [0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,1]$

$C_2 = [1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,0]$ $C_{12} = [0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1]$

$C_3 = [0\,1\,0\,0\,1\,1\,1\,0\,1\,1\,0\,0\,1]$ $C_{13} = [0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0]$

$C_4 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0]$ $C_{14} = [1\,0\,0\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0]$

$C_5 = [1\,0\,1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,0]$ $C_{15} = [1\,1\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,1]$

$C_6 = [0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0\,1\,0]$ $C_{16} = [1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1]$

$C_7 = [0\,1\,1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1]$ $C_{17} = [1\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1]$

$C_8 = [1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1]$ $C_{18} = [1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0]$

$C_9 = [1\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1\,0]$ $C_{19} = [0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,1\,1\,0]$

$C_{10} = [1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,1\,1]$ $C_{20} = [1\,0\,0\,1\,0\,1\,0\,0\,1]$

Selanjutnya, setiap blok *decoding* akan dilakukan operasi perkalian menggunakan kunci publik untuk dibandingkan dengan hasil dekripsi *ciphertext* yang telah dihitung sebelumnya. Kunci publik diperoleh melalui proses membangkitkan

kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter (13,9) yaitu

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{dan } S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai pesan (m), dengan transpose matriks *parity-check* (H^T) ditambah dengan invers dari matriks pengacak (S)

$$H^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{dan } S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Selain itu, proses ini melibatkan vektor sindrom h berukuran $(n-k)$-bit, yaitu $h = [0\ 0\ 0\ 1]$ yang ditambahkan untuk memulihkan *noise* yang diberikan. Hasil perhitungan tersebut sebagai berikut:

$$c'_{(n-k)-bit} = m_{n-bit} H^T_{n\times(n-k)} + h_{(n-k)-bit} S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = m_{13-bit} H^T_{13\times4} + h_{4-bit} S^{-1}_{4\times4}$$

$$c_1 = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c_2 = [1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c_3 = [0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$$

$$c_4 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c_5 = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c_6 = [0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$$

$$c_7 = [0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$$

$$c_8 = [1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$$

$$c_9 = [1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 0\ 1]$$

$$c_{10} = [1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c_{11} = [0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c_{12} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c_{13} = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c_{14} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c_{15} = [1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c_{16} = [1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c_{17} = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$$

$$c_{18} = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

$$c_{19} = [0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

Karena $C_{20} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1]$ tidak memenuhi ukuran 13-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $C_{20} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$

$$c_{20} = [1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

Berdasarkan hasil *decoding* yang telah dilakukan, diperoleh *decoding* isi dokumen dari *dokumen_sign_key2.pdf* dengan kunci kedua, sebagai berikut:

1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1

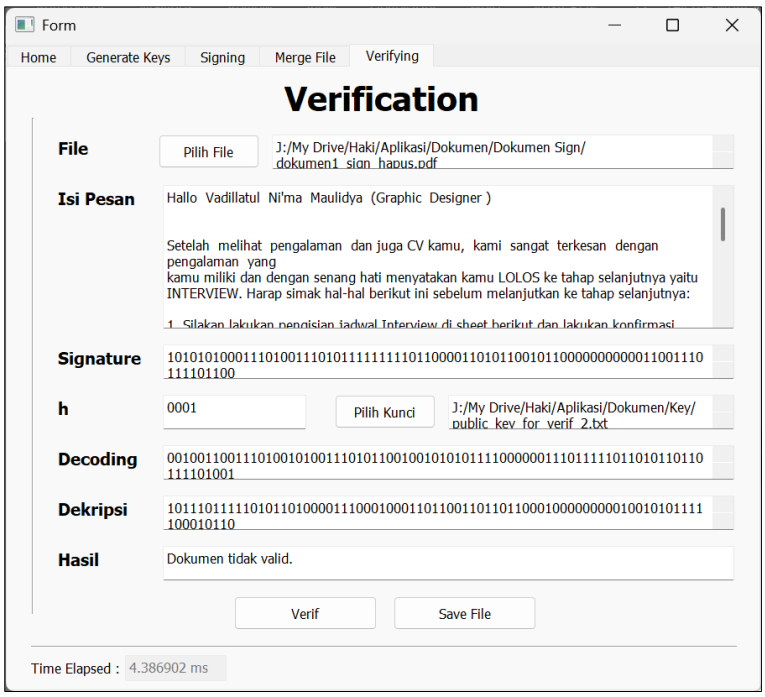
0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0

0 1 0 1 1 0

Berdasarkan perhitungan kedua langkah yang telah dilakukan, diperoleh hasil dekripsi *ciphertext* dan *decoding* isi dokumen menghasilkan nilai yang sama. Dengan demikian, dokumen dengan nama *dokumen1_sign_key2.pdf* yang diterima dianggap valid, hal ini menandakan bahwa dokumen tidak mengalami perubahan selama proses pengiriman.

4.3.3 Menghapus Isi Dokumen

Verifikasi ketiga dilakukan pada dokumen yang telah mengalami modifikasi dengan menghapus sebagian isi dokumen, sebagaimana tercantum dalam Lampiran 6. Selain itu, verifikasi ini dilakukan menggunakan kunci kedua yang telah dibangkitkan sebelumnya, di mana dokumen yang diuji bernama *dokumen1_sign_hapus.pdf*, yang terdiri dari satu halaman. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.9.



Gambar 4.9 Hasil *Verifying* Menghapus Isi Dokumen

Berdasarkan Gambar 4.9, dapat dilihat bahwa hasil dari pemindaian QR Code menghasilkan *digital signature* berupa biner yang sesuai dengan hasil enkripsi pada proses *signing* sebelumnya. Nilai *signature* yang diperoleh dari hasil pemindaian QR Code adalah

1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0

0 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1

1 0 1 1 0 0

Selanjutnya, *signature* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran $(n-k)$ -bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(13,9)$, maka setiap blok akan dibagi menjadi 4-bit. Proses pembagian ini dilakukan untuk mempermudah proses dekripsi memastikan setiap blok (C_i) dari *digital signature* dapat diproses secara terpisah. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\ 0\ 1\ 0]$	$C_8 = [1\ 1\ 1\ 1]$	$C_{15} = [0\ 0\ 0\ 0]$
$C_2 = [1\ 0\ 1\ 0]$	$C_9 = [0\ 1\ 1\ 0]$	$C_{16} = [0\ 0\ 0\ 1]$
$C_3 = [0\ 0\ 1\ 1]$	$C_{10} = [0\ 0\ 0\ 1]$	$C_{17} = [1\ 0\ 0\ 1]$
$C_4 = [1\ 0\ 1\ 0]$	$C_{11} = [1\ 0\ 1\ 0]$	$C_{18} = [1\ 1\ 0\ 1]$
$C_5 = [0\ 1\ 1\ 1]$	$C_{12} = [1\ 1\ 0\ 0]$	$C_{19} = [1\ 1\ 1\ 0]$
$C_6 = [0\ 1\ 0\ 1]$	$C_{13} = [1\ 0\ 1\ 1]$	$C_{20} = [1\ 1\ 0\ 0]$
$C_7 = [1\ 1\ 1\ 1]$	$C_{14} = [0\ 0\ 0\ 0]$	

Selanjutnya, setiap blok *ciphertext* akan didekripsi menggunakan kunci publik yang diperoleh melalui proses membangkitkan kunci yang telah dilakukan

sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter (13,9) yaitu

$$S = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai *ciphertext* (c), dengan invers dari matriks pengacak (S)

$$S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = c_{4-bit} \cdot S^{-1}_{4\times4}$$

$c'_1 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{11} = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$

$c'_2 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{12} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_3 = [0\ 0\ 1\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$ $\quad$ $c'_{13} = [1\ 0\ 1\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$

$c'_4 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{14} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_5 = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$ $\quad$ $c'_{15} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_6 = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$ $\quad$ $c'_{16} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$

$c'_7 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $\quad$ $c'_{17} = [1\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$

$c'_8 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $\quad$ $c'_{18} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$

$c'_9 = [0\ 1\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 1]$ $\quad$ $c'_{19} = [1\ 1\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

$c'_{10} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$ $\quad$ $c'_{20} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

Berdasarkan hasil dekripsi yang telah dilakukan, diperoleh dekripsi *ciphertext* dari *dokumen1_sign_hapus.pdf* dengan kunci kedua, sebagai berikut:

1 0 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1

0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0

0 1 0 1 1 0

Selanjutnya, langkah kedua dalam proses verifikasi yaitu melakukan *decoding* dari isi dokumen *dokumen1_sign_hapus.pdf* yang diterima. Proses *decoding* ini bertujuan untuk mengekstrak informasi asli dari dokumen dan membandingkannya dengan hasil dekripsi *digital signature*. Hasil *decoding* isi dokumen yang diterima diperoleh, sebagai berikut:

1 1 1 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0

0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 1 1 1 1 1

1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1 0 1 0

1 0 0 1 0 1 1 1 1 1 0 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0

0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 0 1 0 1 0 1 1 1 0 0

0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0

1 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 1 1

Selanjutnya, hasil *decoding* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter (13,9), maka setiap blok akan dibagi menjadi 13-bit. Proses pembagian ini dilakukan untuk mempermudah proses *decoding* dari dokumen. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 =$ [1 1 1 1 0 0 1 0 0 0 1 0 0]

$C_2 =$ [1 0 1 0 0 0 0 1 1 1 1 1 0]

$C_3 =$ [1 0 1 1 0 1 1 0 1 0 0 0 1]

$C_4 =$ [1 0 0 0 0 0 0 1 0 0 1 0 0]

$C_5 =$ [1 0 0 1 1 0 0 1 0 1 0 1 1]

$C_6 =$ [0 1 1 0 1 1 1 1 1 1 1 1 1]

$C_7 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0]$ $C_{14} = [1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1]$

$C_8 = [1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0]$ $C_{15} = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0]$

$C_9 = [0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1]$ $C_{16} = [1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1]$

$C_{10} = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1]$ $C_{17} = [1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1]$

$C_{11} = [0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0]$ $C_{18} = [0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1]$

$C_{12} = [0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1]$ $C_{19} = [1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0]$

$C_{13} = [1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1]$ $C_{20} = [1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1]$

Selanjutnya, setiap blok *decoding* akan dilakukan operasi perkalian menggunakan kunci publik untuk dibandingkan dengan hasil dekripsi *ciphertext* yang telah dihitung sebelumnya. Kunci publik diperoleh melalui proses membangkitkan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter $(13,9)$ yaitu

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{dan } S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai pesan (m), dengan transpose matriks *parity-check* (H^T) ditambah dengan invers dari matriks pengacak (S)

$$H^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{dan } S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Selain itu, proses ini melibatkan vektor sindrom h berukuran $(n-k)$-bit, yaitu $h = [0\ 0\ 0\ 1]$ yang ditambahkan untuk memulihkan *noise* yang diberikan. Hasil perhitungan tersebut sebagai berikut:

$$c'_{(n-k)-bit} = m_{n-bit} H^T_{n\times(n-k)} + h_{(n-k)-bit} S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = m_{13-bit} H^T_{13\times4} + h_{4-bit} S^{-1}_{4\times4}$$

$$c'_1 = [1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_2 = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_3 = [1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 1]$$

$$c'_4 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c'_5 = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_6 = [0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 1]$$

$$c'_7 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$$

$$c'_8 = [1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c'_9 = [0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{10} = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_{11} = [0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

$$c'_{12} = [0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c'_{13} = [1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$$

$$c'_{14} = [1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c'_{15} = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$$

$$c'_{16} = [1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 0\ 1]$$

$$c'_{17} = [1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{18} = [0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 0\ 1]$$

$$c'_{19} = [1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$$

Karena $C_{20} = [1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1]$ tidak memenuhi ukuran 13-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $C_{20} = [1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0]$

$$c'_{20} = [1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

Berdasarkan hasil *decoding* yang telah dilakukan, diperoleh *decoding* isi dokumen dari *dokumen1_sign_hapus.pdf* dengan kunci kedua, sebagai berikut

0 0 1 0 0 1 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0 1 0 0 1 0 1

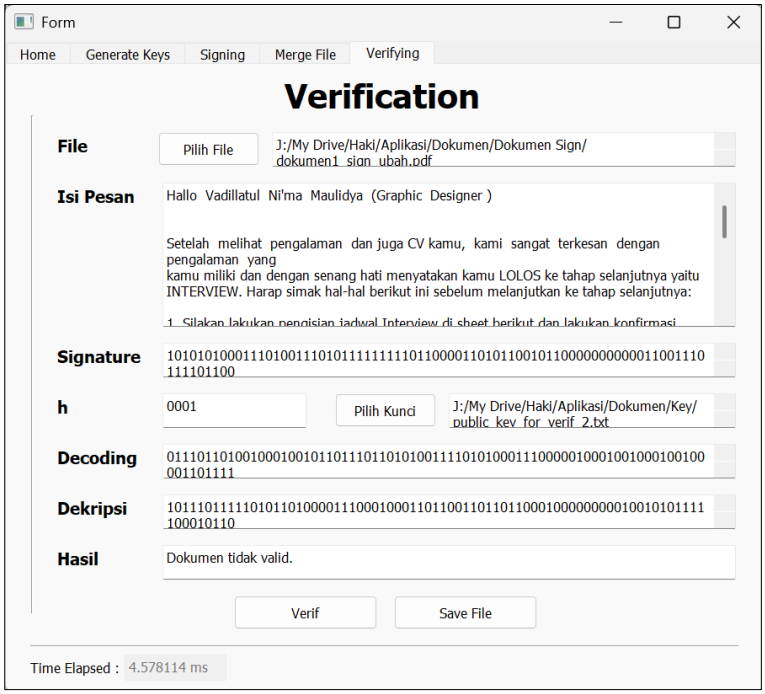
0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1

1 0 1 0 0 1

Berdasarkan perhitungan kedua langkah yang telah dilakukan, diperoleh hasil dekripsi *chipertext* dan *decoding* isi dokumen menghasilkan nilai yang berbeda. Dengan demikian, dokumen dengan nama *dokumen1_sign_hapus.pdf* yang diterima dianggap tidak valid, hal ini menandakan bahwa dokumen mengalami perubahan selama proses pengiriman.

4.3.4 Mengubah Isi Dokumen

Verifikasi keempat dilakukan pada dokumen yang telah mengalami modifikasi dengan mengubah sebagian isi dokumen, sebagaimana tercantum dalam Lampiran 7. Selain itu, verifikasi ini dilakukan menggunakan kunci kedua yang telah dibangkitkan sebelumnya, di mana dokumen yang diuji bernama *dokumen1_sign_ubah.pdf*, yang terdiri dari satu halaman. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.10.



Gambar 4.10 Hasil *Verifying* Mengubah Isi Dokumen

Berdasarkan Gambar 4.10, dapat dilihat bahwa hasil dari pemindaian QR Code menghasilkan *digital signature* berupa biner yang sesuai dengan hasil enkripsi pada proses *signing* sebelumnya. Nilai *signature* yang diperoleh dari hasil pemindaian QR Code adalah

1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0

0 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1

1 0 1 1 0 0

Selanjutnya, *signature* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran $(n-k)$ -bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(13,9)$, maka setiap blok akan dibagi menjadi 4 -bit. Proses pembagian ini dilakukan untuk mempermudah proses dekripsi memastikan setiap blok (C_i) dari *digital signature* dapat diproses

secara terpisah. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\ 0\ 1\ 0]$ $C_8 = [1\ 1\ 1\ 1]$ $C_{15} = [0\ 0\ 0\ 0]$

$C_2 = [1\ 0\ 1\ 0]$ $C_9 = [0\ 1\ 1\ 0]$ $C_{16} = [0\ 0\ 0\ 1]$

$C_3 = [0\ 0\ 1\ 1]$ $C_{10} = [0\ 0\ 0\ 1]$ $C_{17} = [1\ 0\ 0\ 1]$

$C_4 = [1\ 0\ 1\ 0]$ $C_{11} = [1\ 0\ 1\ 0]$ $C_{18} = [1\ 1\ 0\ 1]$

$C_5 = [0\ 1\ 1\ 1]$ $C_{12} = [1\ 1\ 0\ 0]$ $C_{19} = [1\ 1\ 1\ 0]$

$C_6 = [0\ 1\ 0\ 1]$ $C_{13} = [1\ 0\ 1\ 1]$ $C_{20} = [1\ 1\ 0\ 0]$

$C_7 = [1\ 1\ 1\ 1]$ $C_{14} = [0\ 0\ 0\ 0]$

Selanjutnya, setiap blok *ciphertext* akan didekripsi menggunakan kunci publik yang diperoleh melalui proses membangkitkan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter (13,9) yaitu

$$S = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai *ciphertext* (c), dengan invers dari matriks pengacak (S)

$$S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = c_{4-bit} \cdot S^{-1}_{4\times4}$$

$c'_1 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $c'_2 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$

$c'_3 = [0\ 0\ 1\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$ $c'_{12} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_4 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $c'_{13} = [1\ 0\ 1\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$

$c'_5 = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$ $c'_{14} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_6 = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$ $c'_{15} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_7 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $c'_{16} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$

$c'_8 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $c'_{17} = [1\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$

$c'_9 = [0\ 1\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 1]$ $c'_{18} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$

$c'_{10} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$ $c'_{19} = [1\ 1\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

$c'_{11} = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $c'_{20} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

Berdasarkan hasil dekripsi yang telah dilakukan, diperoleh dekripsi *ciphertext* dari *dokumen1_sign_ubah.pdf* dengan kunci kedua, sebagai berikut:

1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1

0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0

0 1 0 1 1 0

Selanjutnya, langkah kedua dalam proses verifikasi yaitu melakukan *decoding* dari isi dokumen *dokumen1_sign_ubah.pdf* yang diterima. Proses *decoding* ini bertujuan untuk mengekstrak informasi asli dari dokumen dan membandingkannya dengan hasil dekripsi *digital signature*. Hasil *decoding* isi dokumen yang diterima diperoleh, sebagai berikut:

1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 1 0 0 0 1

1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1

0 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1

0 1 1 0 0 1 0 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 1 1 1 0

1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 1 0 1

1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0

0 0 0 1 0 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1

Selanjutnya, hasil *decoding* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(13,9)$, maka setiap blok akan dibagi menjadi 13-bit. Proses pembagian ini dilakukan untuk mempermudah proses *decoding* dari dokumen. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0]$

$C_2 = [1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$

$C_3 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1]$

$C_4 = [1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$

$C_5 = [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1]$

$C_6 = [0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0]$

$C_7 = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0]$

$C_8 = [0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1]$

$C_9 = [0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1]$

$C_{10} = [0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0]$

$C_{11} = [0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$

$C_{12} = [0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0]$

$C_{13} = [0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1]$

$C_{14} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1]$

$C_{15} = [1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1]$

$C_{16} = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1]$

$C_{17} = [1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1]$

$C_{18} = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1]$

$C_{19} = [1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]$

$C_{20} = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$

Selanjutnya, setiap blok *decoding* akan dilakukan operasi perkalian menggunakan kunci publik untuk dibandingkan dengan hasil dekripsi *ciphertext* yang telah dihitung sebelumnya. Kunci publik diperoleh melalui proses membangkitkan

kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter (13,9) yaitu

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{dan } S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai pesan (m), dengan transpose matriks *parity-check* (H^T) ditambah dengan invers dari matriks pengacak (S)

$$H^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{dan } S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Selain itu, proses ini melibatkan vektor sindrom h berukuran $(n-k)$-bit, yaitu $h = [0\ 0\ 0\ 1]$ yang ditambahkan untuk memulihkan *noise* yang diberikan. Hasil perhitungan tersebut sebagai berikut:

$$c'_{(n-k)-bit} = m_{n-bit} H^T_{n\times(n-k)} + h_{(n-k)-bit} S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = m_{13-bit} H^T_{13\times4} + h_{4-bit} S^{-1}_{4\times4}$$

$$c'_1 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 1]$$

$$c'_2 = [1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_3 = [1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c'_4 = [1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

$$c'_5 = [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_6 = [0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 0\ 1]$$

$$c'_7 = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 0\ 1]$$

$$c'_8 = [0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_9 = [0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c'_{10} = [0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$$

$$c'_{11} = [0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 0]$$

$$c'_{12} = [0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$$

$$c'_{13} = [0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$$

$$c'_{14} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{15} = [1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{16} = [1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c'_{17} = [1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$$

$$c'_{18} = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$$

$$c'_{19} = [1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

Karena $C_{20} = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$ tidak memenuhi ukuran 13-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $C_{20} = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$

$$c'_{20} = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

Berdasarkan hasil *decoding* yang telah dilakukan, diperoleh *decoding* isi dokumen dari *dokumen1_sign_ubah.pdf* dengan kunci kedua, sebagai berikut

0 1 1 1 0 1 1 0 1 0 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 0 0 1 1

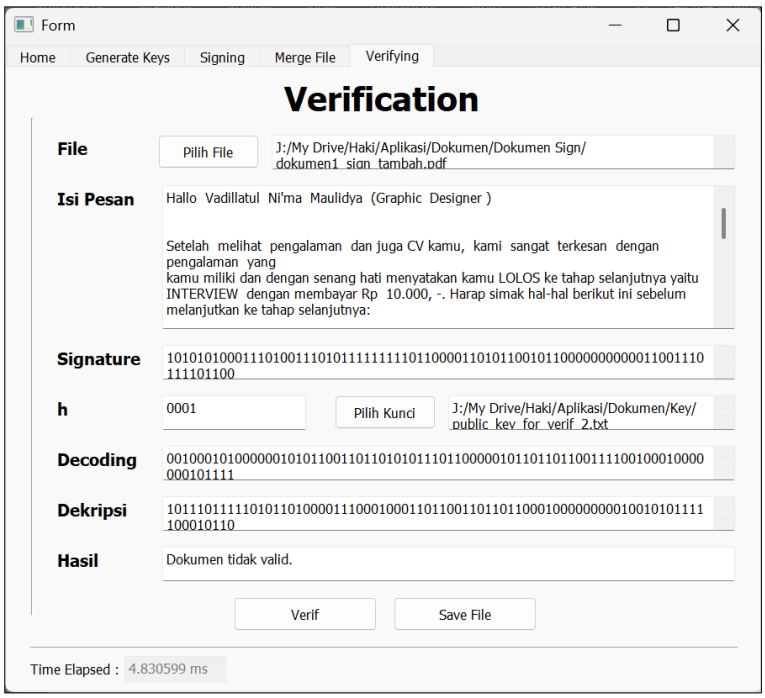
1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1

1 0 1 1 1 1

Berdasarkan perhitungan kedua langkah yang telah dilakukan, diperoleh hasil dekripsi *chipertext* dan *decoding* isi dokumen menghasilkan nilai yang berbeda. Dengan demikian, dokumen dengan nama *dokumen1_sign_ubah.pdf* yang diterima dianggap tidak valid, hal ini menandakan bahwa dokumen mengalami perubahan selama proses pengiriman.

4.3.5 Menambah Isi Dokumen

Verifikasi kelima dilakukan pada dokumen yang telah mengalami modifikasi dengan mengubah sebagian isi dokumen, sebagaimana tercantum dalam Lampiran 8. Selain itu, verifikasi ini dilakukan dengan menggunakan kunci kedua yang telah dibangkitkan sebelumnya, di mana dokumen yang diuji bernama *dokumen1_sign_tambah.pdf*, yang terdiri dari satu halaman. Berdasarkan pengujian yang telah dilakukan menggunakan bantuan program Python, diperoleh hasil sebagaimana ditunjukkan pada Gambar 4.11.



Gambar 4.11 Hasil *Verifying* Menambah Isi Dokumen

Berdasarkan Gambar 4.9, dapat dilihat bahwa hasil dari pemindaian QR Code menghasilkan *digital signature* berupa biner yang sesuai dengan hasil enkripsi pada proses *signing* sebelumnya. Nilai *signature* yang diperoleh dari hasil pemindaian QR Code adalah

1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0

0 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1 1

1 0 1 1 0 0

Selanjutnya, *signature* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran $(n-k)$ -bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter $(13,9)$, maka setiap blok akan dibagi menjadi 4-bit. Proses pembagian ini dilakukan untuk mempermudah proses dekripsi memastikan setiap blok (C_i) dari *digital signature* dapat diproses secara terpisah. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [1\ 0\ 1\ 0]$ $C_8 = [1\ 1\ 1\ 1]$ $C_{15} = [0\ 0\ 0\ 0]$

$C_2 = [1\ 0\ 1\ 0]$ $C_9 = [0\ 1\ 1\ 0]$ $C_{16} = [0\ 0\ 0\ 1]$

$C_3 = [0\ 0\ 1\ 1]$ $C_{10} = [0\ 0\ 0\ 1]$ $C_{17} = [1\ 0\ 0\ 1]$

$C_4 = [1\ 0\ 1\ 0]$ $C_{11} = [1\ 0\ 1\ 0]$ $C_{18} = [1\ 1\ 0\ 1]$

$C_5 = [0\ 1\ 1\ 1]$ $C_{12} = [1\ 1\ 0\ 0]$ $C_{19} = [1\ 1\ 1\ 0]$

$C_6 = [0\ 1\ 0\ 1]$ $C_{13} = [1\ 0\ 1\ 1]$ $C_{20} = [1\ 1\ 0\ 0]$

$C_7 = [1\ 1\ 1\ 1]$ $C_{14} = [0\ 0\ 0\ 0]$

Selanjutnya, setiap blok *ciphertext* akan didekripsi menggunakan kunci publik yang diperoleh melalui proses membangkitkan kunci yang telah dilakukan

sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter (13,9) yaitu

$$S = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai *ciphertext* (c), dengan invers dari matriks pengacak (S)

$$S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Hasil perhitungan tersebut diperoleh sebagai berikut:

$$c'_{(n-k)-bit} = c_{(n-k)-bit} \cdot S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = c_{4-bit} \cdot S^{-1}_{4\times4}$$

$c'_1 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{11} = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$

$c'_2 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{12} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

$c'_3 = [0\ 0\ 1\ 1] \cdot S^{-1} = [1\ 1\ 1\ 0]$ $\quad$ $c'_{13} = [1\ 0\ 1\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$

$c'_4 = [1\ 0\ 1\ 0] \cdot S^{-1} = [1\ 0\ 1\ 1]$ $\quad$ $c'_{14} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_5 = [0\ 1\ 1\ 1] \cdot S^{-1} = [0\ 1\ 0\ 0]$ $\quad$ $c'_{15} = [0\ 0\ 0\ 0] \cdot S^{-1} = [0\ 0\ 0\ 0]$

$c'_6 = [0\ 1\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 1]$ $\quad$ $c'_{16} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$

$c'_7 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $\quad$ $c'_{17} = [1\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$

$c'_8 = [1\ 1\ 1\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$ $\quad$ $c'_{18} = [1\ 1\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$

$c'_9 = [0\ 1\ 1\ 0] \cdot S^{-1} = [1\ 1\ 0\ 1]$ $\quad$ $c'_{19} = [1\ 1\ 1\ 0] \cdot S^{-1} = [0\ 0\ 0\ 1]$

$c'_{10} = [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$ $\quad$ $c'_{20} = [1\ 1\ 0\ 0] \cdot S^{-1} = [0\ 1\ 1\ 0]$

Berdasarkan hasil dekripsi yang telah dilakukan, diperoleh dekripsi *ciphertext* dari *dokumen1_sign_tambah.pdf* dengan kunci kedua, sebagai berikut:

1 0 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1

0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0

0 1 0 1 1 0

Selanjutnya, langkah kedua dalam proses verifikasi yaitu melakukan *decoding* dari isi dokumen *dokumen1_sign_tambah.pdf* yang diterima. Proses *decoding* ini bertujuan untuk mengekstrak informasi asli dari dokumen dan membandingkannya dengan hasil dekripsi *digital signature*. Hasil *decoding* isi dokumen yang diterima diperoleh, sebagai berikut:

0 1 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 0 0 1 0 1 1

1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0

0 0 0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 0

1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1

0 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0

1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1

0 1 0 0 1 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0

Selanjutnya, hasil *decoding* yang diperoleh akan dibagi menjadi beberapa blok (C_i) yang masing-masing berukuran n-bit. Karena pada pengujian ini menggunakan kunci pertama dengan parameter (13,9), maka setiap blok akan dibagi menjadi 13-bit. Proses pembagian ini dilakukan untuk mempermudah proses *decoding* dari dokumen. Proses pembagian blok (C_i) menghasilkan blok-blok sebagai berikut:

$C_1 = [0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0]$ $C_4 = [0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1]$

$C_2 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1]$ $C_5 = [0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0]$

$C_3 = [1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1]$ $C_6 = [0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1]$

$C_7 = [0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1]$

$C_8 = [1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,0\,0]$

$C_9 = [1\,1\,0\,1\,0\,0\,0\,1\,1\,0\,1\,1\,0]$

$C_{10} = [0\,1\,0\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1]$

$C_{11} = [0\,1\,0\,1\,1\,1\,1\,1\,1\,0\,1\,1\,1]$

$C_{12} = [1\,1\,0\,1\,1\,0\,1\,1\,0\,1\,0\,1\,0]$

$C_{13} = [0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,1\,1\,1]$

$C_{14} = [1\,1\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0]$

$C_{15} = [0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,0\,0\,0]$

$C_{16} = [1\,1\,1\,0\,1\,0\,1\,0\,1\,1\,0\,1\,1]$

$C_{17} = [0\,1\,1\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0]$

$C_{18} = [1\,0\,1\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0]$

$C_{19} = [1\,1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1]$

$C_{20} = [1\,1\,0\,0\,0\,0\,0\,0\,0]$

Selanjutnya, setiap blok *decoding* akan dilakukan operasi perkalian menggunakan kunci publik untuk dibandingkan dengan hasil dekripsi *ciphertext* yang telah dihitung sebelumnya. Kunci publik diperoleh melalui proses membangkitkan kunci yang telah dilakukan sebelumnya. Berdasarkan Gambar 4.2, kunci publik yang diperoleh dengan parameter $(13,9)$ yaitu

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ dan } S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dilakukan perkalian antara hasil pembagian blok (C_i) yang dapat disebut sebagai pesan (m), dengan transpose matriks *parity-check* (H^T) ditambah dengan invers dari matriks pengacak (S)

$$H^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ dan } S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Selain itu, proses ini melibatkan vektor sindrom h berukuran $(n-k)$-bit, yaitu $h = [0\ 0\ 0\ 1]$ yang ditambahkan untuk memulihkan *noise* yang diberikan. Hasil perhitungan tersebut sebagai berikut:

$$c'_{(n-k)-bit} = m_{n-bit} H^T_{n\times(n-k)} + h_{(n-k)-bit} S^{-1}_{(n-k)\times(n-k)}$$

$$c'_{4-bit} = m_{13-bit} H^T_{13\times4} + h_{4-bit} S^{-1}_{4\times4}$$

$$c'_1 = [0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_2 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_3 = [1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 0]$$

$$c'_4 = [0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 1]$$

$$c'_5 = [0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$$

$$c'_6 = [0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 0\ 1]$$

$$c'_7 = [0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c'_8 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 0\ 1]$$

$$c'_9 = [1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 1]$$

$$c'_{10} = [0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{11} = [0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c'_{12} = [1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 0\ 1\ 1]$$

$$c'_{13} = [0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 1\ 1\ 0]$$

$$c'_{14} = [1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 0\ 0]$$

$$c'_{15} = [0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

$$c'_{16} = [1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{17} = [0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

$$c'_{18} = [1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 0\ 0]$$

$$c'_{19} = [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [0\ 0\ 1\ 0]$$

Karena $C_{20} = [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ tidak memenuhi ukuran 13-bit, maka akan dilakukan proses padding dengan menambahkan bit nol di akhir agar struktur data tidak mengubah *digital signature* yang diperoleh, sehingga menjadi $C_{20} = [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

$$c'_{20} = [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \cdot H^T + [0\ 0\ 0\ 1] \cdot S^{-1} = [1\ 1\ 1\ 1]$$

Berdasarkan hasil *decoding* yang telah dilakukan, diperoleh *decoding* isi dokumen dari *dokumen1_sign_tambah.pdf* dengan kunci kedua, sebagai berikut

0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 1 0 1 1 1 0

1 1 0 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0

1 0 1 1 1 1

Berdasarkan perhitungan kedua langkah yang telah dilakukan, diperoleh hasil dekripsi *chipertext* dan *decoding* isi dokumen menghasilkan nilai yang berbeda. Dengan demikian, dokumen dengan nama *dokumen1_sign_tambah.pdf* yang diterima dianggap tidak valid, hal ini menandakan bahwa dokumen mengalami perubahan selama proses pengiriman.

Berdasarkan hasil dari proses verifikasi yang dilakukan terhadap dokumen yang sama namun menggunakan dua kunci berbeda menunjukkan hasil verifikasi yang valid meskipun memiliki *digital signature* yang jauh berbeda. Selain itu, pada dokumen yang telah mengalami beberapa modifikasi, seperti menghapus, mengubah, dan menambah sebagian isi dokumen, program juga mampu mendeteksi setiap perubahan yang mengakibatkan ketidaksesuaian hasil verifikasi. Perubahan pada dokumen tersebut berdampak langsung terhadap hasil verifikasi, karena

message digest yang dihasilkan menjadi tidak sesuai dengan *digital signature* awal. Hasil dari kelima pengujian yang telah dilakukan dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil *Verifying* Menggunakan Satu Dokumen yang Sama

Nama Dokumen	Perhitungan V*erifying*	Hasil
dokumen1_sign_key1.pdf	**Dekripsi:** 10110111111101100111000111101110110010100010011001010000110001101001000010100110010011000100001000011101000100111000010010101100001101101010111100010 ***Decoding:*** 10110111111101100111000111101110110010100010011001010000110001101001000010100110010011000100001000011101000100111000010010101100001101101010111100010	Valid
dokumen1_sign_key2.pdf	**Dekripsi:** 10111011111010110100001110001000110110011011000100000000010010101111100010110 ***Decoding:*** 10111011111010110100001110001000110110011011000100000000010010101111100010110	Valid
dokumen1_sign_hapus.pdf	**Dekripsi:** 10111011111010110100001110001000110110011011000100000000010010101111100010110 ***Decoding:*** 00100110011101001010011101011001001010101111000000111011111011010110110111101001	Tidak valid
dokumen1_sign_ubah.pdf	**Dekripsi:** 10111011111010110100001110001000110110011011000100000000010010101111100010110 ***Decoding:*** 01110110100100010010110111011010100111101010001110000010001001000100100001101111	Tidak valid
dokumen1_sign_tambah	**Dekripsi:** 10111011111010110100001110001000110110011011000100000000010010101111100010110	Tidak valid

	Decoding: 00100010100000010101100110110101011101100000101101101100111100100010000000101111	

Hasil pengujian terhadap dokumen yang menggunakan dua kunci berbeda menunjukkan bahwa program memiliki kemampuan yang baik dalam membedakan hasil verifikasi berdasarkan variasi kunci yang diterapkan. Program mampu secara konsisten memverifikasi dokumen dengan benar selama proses decoding dilakukan menggunakan pasangan kunci publik dan privat yang sesuai. Dengan demikian, perbedaan kunci berpengaruh terhadap hasil *encoding*, namun tidak memengaruhi keabsahan hasil *decoding* selama proses verifikasi dilakukan dengan pasangan kunci yang sesuai. Hasil pengujian juga menunjukkan bahwa modifikasi sekecil apa pun yang dilakukan pada isi dokumen, seperti menghapus, menambah, maupun mengubah juga memengaruhi hasil verifikasi. Hal ini menunjukkan bahwa program mampu mendeteksi perubahan secara akurat, serta menjaga integritas dan keaslian dokumen melalui proses *hashing* yang sensitif terhadap bentuk modifikasi.

Adapun hasil pengujian pada proses *verifying* dokumen dari dua puluh dokumen yang dapat dilihat pada Tabel 4.4. Tabel tersebut menyajikan kunci publik, hasil verifikasi yang diperoleh dari proses dekripsi *digital signature*, dan *decoding* isi dokumen. Informasi tersebut digunakan untuk menentukan keaslian dan integritas setiap dokumen yang telah diterima oleh pihak penerima. Dengan demikian, sistem ini tidak hanya mendukung proses otentikasi yang andal, tetapi juga memberikan jaminan atas ketepatan, keamanan, dan kepercayaan terhadap dokumen digital yang dipertukarkan.

Tabel 4.4 Hasil *Verifying* dari Dua Puluh Dokumen

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
1	*dokumen1_sign_key1*	$ac25ff13b2002ade1$ $309c43777a2d439d$ $3668090324b4313$ $3707a1efbd97ed29$	$H^T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$	***Digital Signature:*** 00011001011001011001110010101010001 10100100101101100001000010100110000011 01010101111000011100101100010011001111 01111011111011000111001011101011011100 **Dekripsi:** 10110111111101100111000111101110110001 01000100110010100001100011010010000010 10011001001100010000100001110100010001 11000010010101100001101101010111100001 ***Decoding:*** 10110111111101100111000111101110110001 01000100110010100001100011010010000010 10011001001100010000100001110100010001 11000010010101100001101101010111100001	Valid
	dokumen1_sign_key2	$ac25ff13b2002ade1$ $309c43777a2d439d$ $3668090324b4313$ $3707a1efbd97ed29$		***Digital Signature:*** 10101010001110100111010111111111101100 00110101100101100000000000011001110111 101100 **Dekripsi:** 10111011111010110100001110001000110110 00110110110001000000000100101011111000 010110	

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
			$H^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$	***Decoding:*** 10111011111010110100001110001000110110011010110110001000000000100101011111000101110	Valid
	dokumen1_sign_hapus	$f2250fada3024995b7$ $ffdf9ba6352f8f5$ $ffc457ef2f0ae271$ $1b3b7cbc6f97137$		***Digital Signature:*** 10101010001110100111010111111111011000011010110100101100000000000011001110111101100 **Dekripsi:** 10111011111010110100001110001000110110011010110110001000000000100101011111000101110 ***Decoding:*** 00100110011101001010011101011001001010101111000000111011111011010110110111101001	Tidak valid
	dokumen1_sign_ubah	$f7f6c23d8f9ffc8d9$ $cd3c4863b76cb6b9$ $d04ebc13ca036f67$ $33bb5e845e95009$		***Digital Signature:*** 10101010001110100111010111111111011000011010110100101100000000000011001110111101100 **Dekripsi:** 10111011111010110100001110001000110110011010110110001000000000100101011111000101110 ***Decoding:*** 01110110100100010010110111011010100111101010001110000010001001000100100001101111	Tidak valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
	dokumen1_sign_tambah	$7a97fb765e68f62a1585b6bff4d1b25657efb6a7bfe7b0771d5b7ed53db01b80$		***Digital Signature:*** 101010100011101001110101111111101100001101011001011000000000000110011101111101100 **Dekripsi:** 101110111110101101000011100010001101100110110110001000000000100101011111000010110 ***Decoding:*** 001000101000000101011001101101010111011000001011011011001111001000100000000101111	Tidak valid
2	*dokumen2_sign*	$5e07e7e7ceaa3b6122a38d857ce4ff88e43018e3f6748750838ae9914bc98db7$		***Digital Signature:*** 001000110010110011001011101100001100101101010111111000110111111001101110011010001011010010001010100000000000011100001111101110011000001001001111010 **Dekripsi:** 001010010010000100010101010100000001010110001000011100101000001101000011010110110111011010110110100000000100111000100101000010001101100100100111 0 ***Decoding:*** 101111111011100111101010100110110000000110001100011111111111001111000000010510101011100000111000011010010110010110011011111110101100010101000101101000	Tidak valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
3	*dokumen3_sign*	$828a4336cf769c811$ $50243d60142a51b37$ $9e514f32e6aefa89$ $20e1c0854ada32$	$H^T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$	***Digital Signature:*** 00001011000111010101101111100111000 11 00110010111100100111010000010001101 10 11010110101000111011101100011101100 11 11000011101110000111111010011000101 01 **Dekripsi:** 00000101101110100110010100110100101 10 11101110100011110011110000011001010 01 01011011101100001100110001010011110 10 01100010001000000100010101001101101 10 ***Decoding:*** 10000110000100110110110111100110100 11 10001000000010001011100101111111111 01 01111001011100010000100100111100101 10 00000110110101001101111110110011110 10	Tidak valid
4	*dokumen4_sign*	$a363a1c2d484e24$ $dd53940013bbb67c$ $af4cb119f3baa633$ $cd18d1be6f073de9c$		***Digital Signature:*** 10001011111001010110000110100111000 0100 01011101010101011010000000111101011 11 10101110110110101000011111001001110 11 10010011110011011011001011000110111 10 **Dekripsi:** 11000101001101101100111101110001001 00 01000111110111010100000101100110101 00 11010101011111111010111000001001000 10 00010010001110101111101101100101000 11 ***Decoding:*** 11000101001101101100111101110001001 00 01000111110111010100000101100110101 00	Valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
				11010101011111111010111000001001000100010010001110101111101101100101000110001001000111010111111011011001010001100	
5	*dokumen5_sign*	$0cf126428721838860e6faf5264fa9e559692f9b412bbfcd95f2212364f9aaff$		***Digital Signature:*** 10011110001000011010000001011101001111001001010100010100111000011010010100101001100110011010001000100011001111100100111110111 **Dekripsi:** 00000100011010001100100100100111111111100000111010101001110110001110011000011111000001111100110011000111100000100101010001011100010 ***Decoding:*** 01011101101010101111110000110100101110100011111001100111001110000101000111011011101000001110110110101110001011111110111101100100010	Tidak valid
6	*dokumen6_sign*	$fb938bcd80ac28a15de8824779f8b5fff63b96be9c0033e5cd01425ec6753638$		***Digital Signature:*** 11001110100110111011110010001110001100001011110101101001011001110001010111000001000010001100111000110101011010100001111101111110111 **Dekripsi:** 00111100001001000101100110001101111001110101111111100110110011001010100100010000001111110011110011000111110100111111100011100010 ***Decoding:***	Valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
			$H^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$	00111100001001000101100110001101111001111010111111110011011001100101001000100000011111100111100110001111101001111111100011100010	
7	*dokumen7_sign*	$ff494ea8eedec53d$ $77021cb3d3617dc2$ $e00a5737ba8c7de$ $36327a568b9531a53$		***Digital Signature:*** 01100011011010110001100101000011111000001011010110000110011100010110001100100011110001001111100100100111100001010000010101 **Dekripsi:** 11101010110101011011000010000100011110000011000001011001001010000111010100111010111001101010111110100110000000001001010101111011000101 ***Decoding:*** 11101010110101011011000010000100011110000011000001011001001010000111010100111010111001101010111110100110000000001001010101111011000101	Valid
8	*dokumen8_sign*	$34032b03846e3f$ $1e9a117d7757c2a4$ $4b1479bb76864a4f$ $12f6517f42af3bdcb5$		***Digital Signature:*** 11000110110001111010011110010000110111011111101110001010111001011011001100011000011110001111000110100010001000111100101001010 **Dekripsi:** 00000011000001011101101000101110100000001111101101010110101100001110101101101001101001111101111111101001011011	

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
				10001110001100010 ***Decoding:*** 00000011000001011101101000101110100000 00011111011101010110101100001110101101 10110011010011111011111111010010110110 10001110001100010	
9	*dokumen9_sign*	$0bc091411d8656887c$ $4a7062ccdf7c78e73f$ $c0e23f279704d11887$ $827383c4db$	$H^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$	***Digital Signature:*** 11111111010000011000100101101111011100 00111001100111011010111100101110001001 00010100111111011100010101000111101110 10101001000110011 **Dekripsi:** 10010111101101011110001000011100011111 01100000001101110011110010001111101111 00110000110100111110011011010000001110 01100110100010110 ***Decoding:*** 00110110010101011000010011010111110010 00101000111100000101001000011011010110 01010101001010100100011010000111101 00000111100001011	Tidak valid
10	*dokumen10_sign*	$34b0941c223f778$ $f326cbe5188da429$ $5f80b8c369fee79c02$ $87735fac838832f$		***Digital Signature:*** 00110010001100001100111111010101101101 01111101111001010011011111101100001100 11101101001100101011100000000100001100 11110010010001100 **Dekripsi:** 01000000110010000101111110000010011001	Tidak valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
				0100011000110100001001000100001000010 0011111010110010000100100010000110100 10111011001100010 ***Decoding:*** 0010101100111000000101010101100101001 0110001100110010111111100001100011000 0001010011111100101001001001000010111 10101001101100010	
11	*dokumen11_sign*	$825713a6ee66356c0$ $1047eef558be65578$ $829a3adb266714b$ $5cf12c6339e15c0$	$H^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$	***Digital Signature:*** 1011011101011001000111101010111111101 1010010010100011101001111101100001011 1101100010001110110101100110011110111 10100111011110111 **Dekripsi:** 1110110110001010011000011101001010111 0011110111010111111000011110100011001 0110001011000001100101010001000011000 10010010000001110 ***Decoding:*** 0011000011111000110000001011100011101 0110110001110111101110100111011010111 1100000001110010110000010100111110101 11100011000001110	Tidak valid
12	*dokumen12_sign*	$51cd164131ad476c$ $df7cf8d119a3a838$ $c4edf6651763945a29$ $bff15c8f853b59$		***Digital Signature:*** 0001111100011010110110101011000101010 1000101011101110011110011011011000000 0101011010011101010000011101010000000 10011001010001100	Valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
			$H^T = \begin{bmatrix} 0&0&0&1&0&0 \\ 1&0&0&0&1&1 \\ 0&1&0&0&1&0 \\ 1&0&0&1&1&1 \\ 1&0&1&0&1&0 \\ 0&1&0&0&0&0 \\ 1&0&1&0&0&0 \\ 0&1&0&0&0&0 \\ 1&0&1&1&0&0 \\ 0&0&1&1&0&0 \\ 1&1&0&1&0&1 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{bmatrix}$ dan $S = \begin{bmatrix} 0&0&0&0&0&1&0&1 \\ 0&0&1&1&0&0&1&0 \\ 1&1&1&0&0&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 0&0&0&0&0&1&1&1 \\ 1&0&1&0&1&0&1&1 \\ 0&1&0&1&1&0&1&1 \\ 1&1&0&0&1&0&0&1 \end{bmatrix}$	**Dekripsi:** 00110111101101110111100110110011100100110100111110101110101111110010100100110010010110100001011010101101010110010100100 **Decoding:** 00110111101101110111100110110011100100110100111110101110101111110010100100110010010110100001011010101101010110010100100	
13	*dokumen13_sign*	$4dd92120b41cfb66e863885c5175c0bc89171e556983204cbf13c8d782e487c5$		**Digital Signature:** 01011000010011110101000011111101110101001111101110001101000010111110101101001111101111000011010000101111101011000000101 **Dekripsi:** 11111000000110100100101101110101101011011110100111000010101111101100011000110110111010100110010010 **Decoding:** 01011000100001001100010011010110011111101001111101111001111110110010110001101000101011100000100100101	Tidak valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
14	*dokumen14_sign*	$0c4a13abece3294$ $f8b3c585e594808d$ $ea0aabd3abe6c442$ $37e3b46828b421009$	$H^T = \begin{bmatrix} 0&0&0&1&0&0 \\ 1&0&0&0&1&1 \\ 0&1&0&0&1&0 \\ 1&0&0&1&1&1 \\ 1&0&1&0&1&0 \\ 0&1&0&0&0&0 \\ 1&0&1&0&0&0 \\ 0&1&0&0&0&0 \\ 1&0&1&1&0&0 \\ 0&0&1&1&0&0 \\ 1&1&0&1&0&1 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{bmatrix}$ dan $S = \begin{bmatrix} 0&0&0&0&0&1&0&1 \\ 0&0&1&1&0&0&1&0 \\ 1&1&1&0&0&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 0&0&0&0&0&1&1&1 \\ 1&0&1&0&1&0&1&1 \\ 0&1&0&1&1&0&1&1 \\ 1&1&0&0&1&0&0&1 \end{bmatrix}$	***Digital Signature:*** 01101111101011010100111001101111110100111101110010010101100111011010011111011110000111111111000101 **Dekripsi:** 10111101010011000110110110111110100110010100101110100110110011010011100001010110000011101001001 ***Decoding:*** 10111101010011000110110110111110100110010100101110100110110011010011100001010110000011101001001	Valid
15	*dokumen15_sign*	$ff686e4246483100e$ $43d207b2c7841c42$ $d37dfc45b369a90$ $23e0bab7405fa5c3$		***Digital Signature:*** 00001001110000101100111001110000011100100000011110001110111111011010110100010011011101101000101 **Dekripsi:** 11010001001010010010110101001011110101100011110100111101011010001010011101000000010101101000 1001 ***Decoding:*** 11010001001010010010110101001011110101100011110100111101011010001010011101000000010101101000 1001	Valid
16	*dokumen16_sign*	$d8244428b57e4c5$ $d65f8c414866bdd$ $9357a87f8e7fc0b95$ $244c9f1c96e8e5c06$		***Digital Signature:*** 00100111100000011000110110010110100010000011010111011010111100010111010011100100100100000110110	Tidak valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
				Dekripsi: 01000010000011010101000111101010111 10011110001011100110110000111010100010 00101100101100110011 01 ***Decoding:*** 01101000110110101001011111111111011000 10111011110110111111010111010111111101 01011101100001010011 01	
17	*dokumen17_sign*	$d64dc38988f32850c$ $1590010cce0b65$ $bfe30c9115a0d3d$ $068368457779c217d7$		***Digital Signature:*** 10011000001000101110010000101011111101 0110000100 **Dekripsi:** 11011111011101110001100101111010001111 0111111100 1 ***Decoding:*** 01100110011110000010010001111100000000 1101111110 0	Tidak valid
18	*dokumen18_sign*	$ef1c00dbcc3d953$ $3223cb1a8ad3aa7$ $2b4268fc35bca062$ $4a70ebfa5eadab3733$		***Digital Signature:*** 00001100110011001101101001100000100 01 0011110101 1 **Dekripsi:** 00000110011001100100100011100000111 11 1010001101 0 ***Decoding:*** 00000110011001100100100011100000111 11 1010001101 0	Valid
19	*dokumen19_sign*	$cb5928bde1ca85a$ $1fa24907b294a092$		***Digital Signature:*** 00001010110001011111111000100000000000	Valid

No.	Nama Dokumen	*Message Digest*	Kunci Publik	Perhitungan	Hasil
		$4b7c2419578e16a5$ $c7bfe36323d045e1f$	$H^T = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ dan $S = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$	00000100111 **Dekripsi:** 00001000011010110011000101110000000000001111100 ***Decoding:*** 00001000011010110011000101110000000000001111100	
20	*dokumen20_sign*	$d1325a9fe1b8d661$ $07010414fcabfcd$ $59ccb8efb6a22f78$ $0d6c8a03e82f44903$		***Digital Signature:*** 01011000001111101100100000101101010010000111001 **Dekripsi:** 10111111010100010110111101110100100111111001101 ***Decoding:*** 11011001110011100101111110110011010011111111010	Tidak valid

4.4 Analisis Keamanan Menggunakan Efek Avalanche

Efek Avalanche adalah karakteristik penting dalam algoritma kriptografi yang menunjukkan sejauh mana perubahan kecil pada *input* dapat menyebabkan perubahan besar pada *output*, sehingga mencerminkan tingkat keandalan keamanannya. Efek Avalanche yang ideal adalah ketika perubahan satu bit pada *input* menghasilkan perubahan sekitar 50% bit pada *output*, yang mencerminkan bahwa algoritma tersebut memiliki sifat difusi yang kuat. Pada penelitian ini, evaluasi Efek Avalanche dilakukan menggunakan algoritma *hashing* SHA-3 dengan dua pendekatan, yaitu dengan dan tanpa algoritma McEliece sebagai mekanisme tambahan.

Pendekatan pertama bertujuan untuk melihat sejauh mana Efek Avalanche terjadi hanya pada proses *hashing* dengan membandingkan *output hash* biner dari *input* asli dan *input* yang telah dimodifikasi satu bit. Perubahan jumlah bit pada *output* digunakan untuk menghitung persentase perubahan, yang mencerminkan kekuatan difusi dari algoritma SHA-3 secara mandiri. Pendekatan kedua dilakukan dengan menambahkan algoritma McEliece setelah *hashing* untuk menghasilkan *digital signature* berupa biner. Kemudian, hasil *signature* dibandingkan antara *input* asli dan *input* yang dimodifikasi guna menilai kontribusi algoritma McEliece terhadap sensitivitas *output*. Seperti halnya yang telah dijelaskan pada Kajian Teori, proses perhitungan dilakukan dengan menggunakan persamaan berikut:

$$Avalanche\ Effect\ (\%) = \left(\frac{jumlah\ bit\ yang\ berubah}{total\ bit\ output}\right) \times 100\%$$

Tabel 4.5 merupakan tabel yang menyajikan perbandingan eksekusi yang menggambarkan perbedaan antara kedua pendekatan.

Tabel 4.5 Hasil Pengujian Efek Avalanche Tanpa Tambahan McEliece

Input **Asli**	***Input*** **Diubah**	***Digital Signature***	**Bit Berbeda**	**Persentase**
Input123	Input12	**Asli:** 10010111001100110010010100000 1 10111101011011101100110100010 1 01010111000110110101110000001 0 00110101101111010101100001100 1 11101110011011011001001000010 0 10001011011010010001110001010 0 00110101110011111100011000111 1 00000111011101100110101110110 0 0100000111000110	125/256 bit	48,83%
		Diubah: 01001000010110000110100001011 0 01110111000100101100000101011 1 10101001100010000110000000000 1 01110000100100011010001001100 0 01110000000110101001000101110 1 01100010010010010011110010010 0 11000111011101011111011010110 1 11010111100111010111110111110 1 1000101110110000		
Input123	Input124	**Asli:** 10010111001100110010010100000 1 10111101011011101100110100010 1 01010111000110110101110000001 0 00110101101111010101100001100 1 11101110011011011001001000010 0 10001011011010010001110001010 0 00110101110011111100011000111 1 00000111011101100110101110110 0 0100000111000110 **Diubah:** 01001100100001111101111001111 1 00111111100101001010110101100 1 10000110000001011100010111000 0 01000001011001111100111001011 0 01111001110111111001000101011 0 01000111011000001001100111010 0 01110100011110011000001001110 0 11101011100010010001100101001 1 1001100111111101	133/256 bit	51,95%
Input123	Input1234	**Asli:** 10010111001100110010010100000 1 10111101011011101100110100010 1 01010111000110110101110000001 0 00110101101111010101100001100 1 11101110011011011001001000010 0	135/256 bit	51,56%

		10001011011010010001110001010 00110101110011111100011000111 00000111011101100110101110110 0100000111000110 **Diubah:** 01110111001110011100110100110 01111111111010110101001010111 00100111000100011100101111101 11001100010100101100011001000 01000000010100000011011101100 00001110101101111010000101001 01100001111100010111010101100 00011110100111100010000101111 0 1010100010100110		

Selain itu, pengujian juga dilakukan dengan membandingkan dua kunci enkripsi yang berbeda untuk mengetahui perubahan struktur kunci yang dapat memengaruhi hasil enkripsi. Hal ini, dapat memastikan bahwa algoritma McEliece menghasilkan *output* yang sensitif terhadap variasi kunci. Tabel 4.6 dan Tabel 4.7 merupakan tabel yang menyajikan perbandingan eksekusi yang menggambarkan perbedaan kunci enkripsi.

Tabel 4.6 Hasil Pengujian Efek Avalanche dengan Kunci Pertama

***Input* Asli**	***Input* Diubah**	***Digital Signature***	**Bit Berbeda**	**Persentase**
Input123	Input12	**Asli:** 00101011100101001001000111000 11101011000101110011011011110 00010010100001110111110111000 01000011110100111101010000110 0 01000000011100010101010000 **Diubah:** 00001000111000011001010110000 01010000101101111010001001110 00011011010101000100101100011 11000010100011000000101000001 1 11110111000100101100000110 10	72/148 bit	48,65%

Input123	Input124	**Asli:** 00101011100101001001000111000111101011000101110011011011111000100101000001110111010011110101000011000100000111000101010101000**Diubah:** 00101011100001110111101011001011101100101011100011101111111001110010011100011010011110010001110000111010000001101011110101001011001001101001111011000110	70/148 bit	47,30%
Input123	Input1234	**Asli:** 00101011100101001001000111000111101011000101110011011011111000100101000001110111010011110101000011000100000111000101010101000**Diubah:** 00011001001110101100001110011110100101100100101110010010001010110100110001100010001110100000010000001011111010101110101000111001100010001111010000	77/148 bit	52,03%

Tabel 4.7 Hasil Pengujian Efek Avalanche dengan Kunci Kedua

Input **Asli**	***Input*** **Diubah**	***Digital Signature***	**Bit Berbeda**	**Persentase**
Input123	Input12	**Asli :** 11101010010110110111101001111110101011111111001000000111011001001010110111101110**Diubah :** 00000110000110110111111110101011101011100010010101001010011110000101110111110111011	42/80 bit	52,50%
Input123	Input124	**Asli :** 11101010010110110111101001111110101011111111001000000111011001001010110111101110**Diubah :** 11100000001110010001110000101111010100100011000001001100011011100100000010011111	37/80 bit	46,25%

Input123	Input1234	**Asli :** 11101010010110110111101001111 10101011111110010000001110110 0100101011011110111 00 **Diubah :** 00000110000110110111111101010 11010111000100101010010100111 10000101111110111 0001	41/80 bit	51,25%

Berdasarkan hasil pengujian yang dilakukan dengan dan tanpa tambahan McEliece, dapat dilihat bahwa kombinasi algoritma SHA-3 dengan ditambah algoritma McEliece mampu menghasilkan perubahan signifikan pada *output digital signature* ketika terjadi perubahan kecil pada *input*. Pada Tabel 4.5, menunjukkan bahwa pengujian tanpa tambahan McEliece menghasilkan persentase perubahan bit berkisar antara 48,65% hingga 51,95% dengan rata-rata 50,78% dari total 256 bit. Nilai ini cukup membuktikan standar ideal Efek Avalanche, yaitu sekitar 50% yang menunjukkan bahwa algoritma memiliki karakteristik difusi yang baik.

Sementara itu, pengujian dengan tambahan McEliece menghasilkan persentase perubahan bit sedikit lebih tinggi, hal ini menunjukkan bahwa McEliece memperkuat Efek Avalanche dan memberikan hasil yang sensitif terhadap variasi kunci. Seperti halnya pada Tabel 4.6, menunjukkan bahwa pengujian dengan kunci pertama menunjukkan bahwa perubahan satu bit pada *input* menghasilkan rata-rata perubahan sebesar 49,32% dari total 148 bit *output digital signature*. Sementara itu, pada Tabel 4.7, menunjukkan bahwa pengujian dilakukan dengan menggunakan kunci kedua dengan persentase perubahan berkisar antara 52,50% hingga 46,25% dengan rata-rata 50% dari total 80 bit. Nilai rata-rata yang lebih tinggi dibanding kunci pertama mengindikasikan bahwa pemilihan kunci turut memengaruhi kekuatan Efek Avalanche. Selain itu, pengujian ini menunjukkan bahwa algoritma McEliece baik dengan dan tanpa tambahan enkripsi memiliki ketahanan yang baik

terhadap serangan dengan memperlihatkan kemampuan Efek Avalanche yang mendekati ideal.

4.5 Pengujian Waktu Pemrosesan

Pengujian waktu dalam proses enkripsi dan dekripsi sangat penting dalam menilai efisiensi dan performa algoritma, khususnya dalam implementasi algoritma yang memerlukan *hashing* data secara *real-time*. Pada penelitian ini, pengujian waktu *hashing* dilakukan untuk membandingkan performa algoritma SHA-3 dalam dua pendekatan, yaitu dengan dan tanpa algoritma McEliece sebagai mekanisme tambahan dalam proses enkripsi dan dekripsi. Proses pengujian ini melibatkan enam dokumen dengan ukuran yang beragam untuk mengukur waktu yang dibutuhkan dalam menghasilkan *output* enkripsi dan dekripsi. Setiap pengujian dilakukan sebanyak lima kali untuk setiap dokumen dengan hasil yang dianalisis berdasarkan rata-rata waktu eksekusi. Perbedaan utama antara kedua pendekatan terletak pada penggunaan algoritma McEliece, di mana algoritma ini diimplementasikan pada tahap enkripsi dan dekripsi untuk memberikan lapisan tambahan dan kompleksitas dalam sistem. Algoritma McEliece yang berbasis pada algoritma kriptografi kunci publik dengan kode koreksi kesalahan, sehingga ketika ditambahkan lapisan enkripsi dapat memengaruhi efisiensi waktu pemrosesan secara keseluruhan.

Hasil pengujian menunjukkan perbedaan waktu pemrosesan antara kedua pendekatan tersebut, dengan fokus pada bagaimana kompleksitas tambahan dari algoritma McEliece yang berpengaruh terhadap durasi enkripsi dan dekripsi pada berbagai ukuran dokumen. Dengan membandingkan waktu eksekusi kedua metode,

pengujian ini akan menjawab tujuan dari penelitian, apakah peningkatan keamanan yang diberikan algoritma McEliece sebanding dengan efisiensi waktu yang mungkin terjadi. Tabel 4.8 dan Tabel 4.9 merupakan tabel yang menyajikan perbandingan eksekusi yang menggambarkan perbedaan performa antara kedua pendekatan selama proses enkripsi.

Tabel 4.8 Hasil Pengujian Waktu Pemrosesan Enkripsi Tanpa Tambahan McEliece

Nama Dokumen	**Ukuran *file* (kb)**	**Uji Coba Enkripsi tanpa McEliece (ms)**					
		ke−1	**ke−2**	**ke−3**	**ke−4**	**ke−5**	**rata-rata**
file 1.pdf	12	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
file 2.pdf	63	1×10^{-6}	0,45021	1×10^{-6}	1×10^{-6}	1,91104	0,47225
file 3.pdf	154	1,26862	1×10^{-6}	1×10^{-6}	1,07204	0,68514	0,60516
file 4.pdf	1.315	1×10^{-6}	1,19221	0,50735	1×10^{-6}	1,46269	0,63245
file 5.pdf	5.614	1,07717	1,01065	0,51808	0,99015	1×10^{-6}	0,71921
file 6.pdf	11.231	1,51324	2,04062	1,00827	1,50919	1,51003	1,51627

Tabel 4.9 Hasil Pengujian Waktu Pemrosesan Enkripsi dengan Tambahan McEliece

Nama Dokumen	**Ukuran *file* (kb)**	**Uji Coba Enkripsi dengan McEliece (ms)**					
		ke−1	**ke−2**	**ke−3**	**ke−4**	**ke−5**	**rata-rata**
file 1.pdf	12	2,29430	1,62665	2,00629	1,51293	2,19559	1,92715
file 2.pdf	63	2,87485	1,03037	2,54893	1,65836	2,92253	2,20701
file 3.pdf	154	3,16095	2,70676	1,12812	3,59344	2,01551	2,52096
file 4.pdf	1.315	3,25632	3,02267	1,34072	2,56133	3,01933	2,64007
file 5.pdf	5.614	2,73601	3,58462	2,61561	3,52644	3,74858	3,24225
file 6.pdf	11.231	4,45675	3,07004	3,87883	3,53805	4,51970	3,89267

Pada Tabel 4.8 dan Tabel 4.9, dapat dilihat bahwa penggunaan algoritma McEliece dalam proses enkripsi menyebabkan peningkatan waktu pemrosesan dibandingkan dengan enkripsi tanpa tambahan McEliece. Total waktu yang diperlukan untuk enkripsi dengan McEliece menunjukkan rata-rata yang lebih tinggi dibandingkan dengan enkripsi tanpa McEliece. Sebagai contoh, untuk dokumen terkecil berukuran 12 kb, rata-rata waktu dekripsi tanpa McEliece adalah

1×10^{-6} ms, sedangkan dengan McEliece sebesar 3,601283 ms. Sedangkan, untuk dokumen terbesar (11.231 kb), rata-rata waktu enkripsi tanpa McEliece adalah 1,51627 ms, sedangkan dengan McEliece sebesar 3,89267 ms. Hal ini menunjukkan bahwa algoritma McEliece menambah kompleksitas dalam proses enkripsi. Namun, di sisi lain algoritma McEliece dapat memberikan peningkatan dalam aspek keamanan yang melibatkan operasi matriks dan permutasi bit.

Peningkatan waktu ini menunjukkan adanya *trade-off* antara keamanan dan efisiensi, di mana penambahan algoritma McEliece memperkuat perlindungan dokumen tetapi juga menambah beban komputasi. Selain itu, dokumen yang telah ditandatangani secara digital mengalami peningkatan ukuran akibat tambahan informasi dari proses enkripsi dan *digital signature*. Oleh karena itu, untuk mendapatkan gambaran yang lebih menyeluruh mengenai dampak implementasi algoritma ini, diperlukan analisis lebih lanjut pada tahap dekripsi. Selanjutnya, akan dilakukan proses dekripsi menggunakan dua pendekatan untuk mengevaluasi performa sistem dalam memulihkan dan memverifikasi dokumen yang diterima. Tabel 4.10 dan Tabel 4.11 merupakan tabel yang menyajikan perbandingan eksekusi yang menggambarkan perbedaan performa antara kedua pendekatan selama proses dekripsi.

Tabel 4.10 Hasil Pengujian Waktu Pemrosesan Dekripsi Tanpa Tambahan McEliece

Nama Dokumen	**Ukuran *file* (kb)**	**Uji Coba Dekripsi tanpa McEliece (ms)**					
		ke−1	**ke−2**	**ke−3**	**ke−4**	**ke−5**	**rata-rata**
file 1.pdf	86	0,99468	1×10^{-6}	1×10^{-6}	1×10^{-6}	1,06382	0,41170
file 2.pdf	118	1,08337	1,01280	1,02710	1×10^{-6}	1,00040	0,82473
file 3.pdf	246	1×10^{-6}	1,62291	0,99420	1,44195	0,47366	0,90654
file 4.pdf	1.331	0,52356	1,98793	1,30367	0,99515	0,50990	1,06404
file 5.pdf	5.484	1,30796	2,08401	2,01797	1,51801	1,99585	1,78476
file 6.pdf	10.906	3,12018	3,03983	4,03189	3,00049	3,00693	3,23987

Tabel 4.11 Hasil Pengujian Waktu Pemrosesan Dekripsi dengan Tambahan McEliece

Nama Dokumen	Ukuran *file* (kb)	Uji Coba Dekripsi dengan McEliece (ms)					
		ke−1	ke−2	ke−3	ke−4	ke−5	rata-rata
file 1.pdf	86	4,34875	3,81708	4,23073	2,55696	3,05287	3,60128
file 2.pdf	118	4,58860	5,06997	3,51517	5,04612	2,30607	4,10519
file 3.pdf	246	4,65130	3,60290	2,17113	5,10644	5,10120	4,12660
file 4.pdf	1.331	5,04994	3,06305	5,55753	5,16414	4,29844	4,62662
file 5.pdf	5.484	6,34121	5,52487	4,60743	5,94472	4,94027	5,47170
file 6.pdf	10.906	8,14747	6,69455	5,48911	6,4556	5,70011	6,49738

Sama halnya dengan hasil pengujian waktu pemrosesan enkripsi, berdasarkan hasil pengujian yang ditampilkan pada Tabel 4.10 dan Tabel 4.11, dapat disimpulkan bahwa proses dekripsi dengan tambahan algoritma McEliece membutuhkan waktu yang lebih lama dibandingkan dengan dekripsi tanpa McEliece. Dari enam dokumen PDF yang diuji, terlihat bahwa ukuran dokumen yang lebih besar cenderung membutuhkan waktu pemrosesan dekripsi lebih lama. Sebagai contoh, untuk dokumen terbesar (10.906 kb), rata-rata waktu dekripsi tanpa McEliece adalah 3,239870 ms, sedangkan dengan McEliece sebesar 6,497383 ms. Perbedaan ini menunjukkan bahwa penerapan McEliece menambah *overhead* komputasi yang cukup signifikan dalam proses dekripsi. Selain itu, dokumen yang lebih kecil juga mengalami peningkatan waktu dekripsi saat McEliece diterapkan. Misalnya, untuk dokumen berukuran 86 kb, rata-rata waktu dekripsi tanpa McEliece adalah 0,411701 ms, sedangkan dengan McEliece sebesar 3,601283 ms.

Berdasarkan hasil pengujian ini, dapat diketahui bahwa algoritma McEliece mampu memberikan dampak signifikan terhadap peningkatan waktu enkripsi maupun dekripsi, baik untuk dokumen berukuran besar maupun kecil. Meskipun metode ini memberikan tambahan lapisan keamanan pada proses autentikasi dokumen digital, namun perlu diperhatikan bahwa ada trade-off dalam bentuk

peningkatan beban komputasi. Oleh karena itu, pemanfaatan algoritma McEliece dalam sistem *digital signature* perlu dipertimbangkan secara cermat seperti efisiensi waktu yang sesuai dengan kebutuhan serta kapasitas sistem.

4.6 Integrasi Nilai Keislaman dalam Otentikasi Digital

Perkembangan teknologi digital telah membawa transformasi besar dalam berbagai aspek kehidupan, termasuk dalam hal otentikasi dokumen yang kini tidak lagi terbatas pada tanda tangan manual. Keabsahan dokumen menjadi hal yang sangat krusial dalam menunjang transaksi, komunikasi, hingga kerja sama antar lembaga dan individu di era modern. Namun demikian, tantangan seperti pemalsuan dokumen, manipulasi data, dan pencurian identitas digital menjadi ancaman serius yang tidak bisa diabaikan. Seperti yang telah dijelaskan pada Kajian Teori 2.9, Islam telah mengajarkan umatnya nilai-nilai kejujuran, keadilan, dan amanah dalam setiap interaksi sosial. Di mana konsep ini bersifat universal dalam mengajarkan dasar etis dan spiritual untuk mendukung praktik otentikasi digital yang dapat dipercaya, adil, dan bertanggung jawab. Oleh karena itu, pembahasan ini bertujuan untuk mengaitkan prinsip Islam dengan praktik otentikasi *digital signature*, guna memastikan sistem digital yang diterapkan sejalan dengan ajaran Islam, yaitu keadilan dan amanah.

4.6.1 Keadilan dan Transparansi dalam Transaksi

Islam sangat mengedepankan kejelasan dan transparansi dalam setiap bentuk transaksi, baik bersifat keuangan, perjanjian, maupun administratif. Ajaran Islam menekankan pentingnya pencatatan kesepakatan guna menghindari

perselisihan, kecurangan, maupun kerugian di kemudian hari. Sebagaimana tercantum dalam QS. Al-Baqarah ayat 283 yang menunjukkan bahwa Islam memberikan landasan kuat bagi praktik pencatatan dan verifikasi informasi, bahkan ketika tidak memungkinkan adanya pencatat resmi. Seseorang yang melakukan transaksi baik dalam bentuk jual beli, utang-piutang, atau akad saham yang pembayarannya tidak secara tunai atau ditunda, maka wajib hukumnya untuk membuat surat bukti transaksi yang dilengkapi dengan rincian yang jelas.

Dalam tafsirnya menjelaskan bahwa juru tulis yang berhak mencatat surat tanda bukti haruslah orang yang dapat dipercaya, adil, netral, memahami ilmu fikih, memiliki akhlak yang baik, serta cermat dan cerdas dalam menjalankan tugasnya. Hal ini menunjukkan bahwa dalam Islam otentikasi suatu dokumen tidak hanya terletak pada isinya, tetapi juga pada siapa yang mencatat dan menjamin validitasnya. Dalam dunia digital, peran juru tulis dapat dianalogikan dengan sistem otentikasi yang memiliki fungsi menjaga integritas dan keabsahan dokumen. Sistem otentikasi digital yang dirancang dengan prinsip keadilan, netralitas, serta ketelitian ibarat juru tulis modern yang tidak memihak dan memastikan bahwa data yang tercatat benar adanya, tidak dimanipulasi, dan dapat dipertanggungjawabkan.

Allah menetapkan bahwa pihak yang bertanggung jawab dalam suatu transaksi dalam tafsirnya disebut sebagai pihak penanggung utang adalah pihak yang berhak mendiktekan atau menyampaikan isi dokumen. Hal ini dilakukan karena pihak tersebut yang akan menanggung segala konsekuensi dari perjanjian. Ketika seseorang mendiktekan atau menyampaikan informasi, hal tersebut menjadi bukti yang tidak dapat dibantah atau dipungkiri karena menjadi tanggung

jawab pribadi. Dalam dunia digital, otentikasi berfungsi sebagai sarana yang memastikan setiap informasi yang dibagikan atau disampaikan oleh pihak terkait tercatat dengan jelas dan tidak bisa dimanipulasi. Prinsip ini menguatkan pentingnya menggunakan teknologi otentikasi, seperti *digital signature* yang mampu menjadi bukti sah atas keabsahan dokumen.

4.6.2 Nilai Amanah dalam Islam

Amanah merupakan salah satu nilai moral agung dalam Islam yang mencerminkan kepercayaan, tanggung jawab, dan integritas seseorang dalam menjalankan tugas atau memelihara sesuatu yang dipercayakan kepadanya. Konsep ini tidak hanya berlaku dalam hubungan sosial dan keagamaan, tetapi juga sangat relevan dalam pengelolaan informasi digital dan dokumen. Dalam Al-Qur'an, Allah berfirman dalam QS. An-Nisa ayat 58 yang menegaskan pentingnya menjaga kepercayaan dalam setiap bentuk tanggung jawab. Bentuk amanah atau kepercayaan pada dasarnya tidak hanya berhubungan dengan orang lain, namun dapat pula berhubungan langsung dengan diri sendiri ataupun dengan Allah.

Konsep amanah kepada Allah berarti melaksanakan perintah-Nya, menjauhi larangan-Nya, serta menggunakan seluruh potensi diri untuk mendekatkan diri kepada-Nya. Sementara itu, amanah terhadap diri sendiri meliputi usaha menjaga keselamatan, melakukan hal yang bermanfaat, serta menjauhi tindakan yang merugikan kehidupan dunia maupun akhirat. Sedangkan amanah terhadap sesama mencakup kejujuran dalam transaksi, menjaga rahasia, tidak menipu, serta mengembalikan hak orang lain. Dalam otentikasi digital, nilai amanah sangat relevan untuk semua pihak yang mengelola atau mengakses dokumen digital

dengan tanggung jawab moral untuk menjaga keaslian informasi dan tidak memanipulasi data.

Sejalan dengan nilai amanah dalam Islam, *digital signature* dapat dipandang sebagai bentuk nyata dari amanah di era digital. Teknologi ini berfungsi menjaga keaslian dan integritas dokumen, serta memastikan bahwa informasi yang diterima benar-benar berasal dari pihak yang berwenang dan tidak mengalami perubahan selama proses pengiriman. Dalam praktiknya, *digital signature* dapat mencatat identitas pengirim, waktu pengiriman, serta menjamin bahwa isi dokumen tidak dimanipulasi selama proses transmisi. Oleh karena itu, *digital signature* tidak hanya berperan sebagai alat keamanan, tetapi juga sebagai simbol amanah digital. Selain itu, *digital signature* juga dapat menjadi jaminan bahwa informasi yang dibagikan dilakukan dengan penuh tanggung jawab, kejujuran, dan dapat dipercaya.

4.6.3 Etika dan Tanggung Jawab dalam Penggunaan Teknologi

Teknologi pada dasarnya merupakan alat yang bersifat netral dan dampak dari penggunaannya bergantung pada penggunanya. Dalam Islam, setiap individu diberikan tanggung jawab dalam menggunakan ilmu dan teknologi dengan cara yang sesuai dengan prinsip-prinsip agama. Islam mengajarkan bahwa setiap tindakan harus didasarkan pada niat yang baik dan bertujuan untuk kebaikan, serta memperhatikan dampak yang ditimbulkan terhadap diri sendiri, orang lain, dan lingkungan. Oleh karena itu, ketika seseorang menggunakan teknologi akan dituntut untuk menjunjung tinggi nilai-nilai kejujuran, keadilan, dan amanah, sebagai wujud tanggung jawab atas ilmu dan sarana yang Allah titipkan.

Setiap tindakan digital memiliki konsekuensi moral dan spiritual yang akan dipertanggungjawabkan kelak di hadapan Allah. Dalam hal ini, penggunaan teknologi seperti *digital signature* harus mencerminkan nilai-nilai keislaman seperti kejujuran, tanggung jawab, dan amanah. Dokumen digital yang di otentikasi bukan hanya mewakili transaksi atau komunikasi semata, tetapi juga menjadi representasi dari komitmen dan integritas di hadapan manusia dan Allah. Ketika seseorang menandatangani dokumen secara digital, maka telah dinyatakan bahwa informasi tersebut benar dan dapat dipercaya. Oleh sebab itu, umat Islam dituntut untuk memperlakukan setiap proses digital dengan hati-hati, memastikan bahwa tidak ada manipulasi atau penipuan, dan senantiasa menyadari bahwa setiap amal, sekecil apa pun, akan dicatat dan dimintai pertanggungjawaban.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan uraian pada rumusan masalah dan pembahasan di atas, dapat diambil kesimpulan bahwa:

1. Setiap dokumen digital yang ditandatangani menghasilkan *digital signature* yang berbeda, meskipun menggunakan kunci yang sama. Hal ini terjadi karena *digital signature* bergantung pada nilai *hash* yang dihasilkan dari isi dokumen. Selain itu, penggunaan variasi kunci mampu menghasilkan *digital signature* yang berbeda, meskipun dokumen memiliki konten yang serupa.
2. Dokumen yang tidak mengalami perubahan isi akan menghasilkan nilai dekripsi *digital signature* dan nilai *hash* yang identik saat proses verifikasi. Sebaliknya, jika dokumen mengalami perubahan bahkan hanya satu karakter, nilai dekripsi *digital signature* dan nilai *hash* akan berbeda dari *hash* dokumen asli. Hal ini dikarenakan setiap modifikasi pada isi dokumen akan memengaruhi nilai *hash* yang dihasilkan, sehingga sistem mendeteksi perubahan atau manipulasi secara akurat.
3. Kombinasi algoritma SHA-3 dan McEliece menunjukkan Efek Avalanche yang kuat, di mana perubahan satu bit pada *input* menghasilkan perubahan *output* mendekati 50%. Hal ini menunjukkan bahwa sistem memiliki sifat difusi yang baik. Namun, kompleksitas tinggi pada algoritma McEliece menyebabkan waktu pemrosesan enkripsi dan dekripsi meningkat seiring dengan ukuran dokumen, sehingga efisiensi perlu diperhatikan dalam penerapannya pada dokumen berukuran besar.

Kesimpulan ini menunjukkan bahwa kombinasi algoritma SHA-3 dan McEliece dengan kode Hamming mampu mendeteksi perubahan dokumen dengan akurat. Selain itu, kompleksitas algoritma McEliece memberikan lapisan keamanan, sehingga kombinasi ini tidak hanya meningkatkan keamanan tetapi juga memastikan efisiensi dalam proses otentikasi dokumen digital.

5.2 Saran

Pada penelitian ini, penulis berfokus pada pembuatan *digital signature* menggunakan algoritma SHA-3 dan McEliece pada dokumen berformat .pdf yang hanya berisi alfabet, angka, dan karakter. Hasil penelitian menunjukkan bahwa metode ini efektif dalam menjaga integritas dan mendeteksi perubahan pada dokumen teks. Sehingga untuk penelitian selanjutnya, diharapkan pengembangan lebih lanjut dapat mencakup pembuatan *digital signature* untuk dokumen yang juga mengandung gambar atau tabel. Selain itu, penelitian ini dapat diperluas untuk menerapkan *digital signature* pada format pesan lain, seperti *file* audio (.mp3) dan video (.mp4), guna meningkatkan cakupan dan keamanan otentikasi berbagai jenis data digital.

DAFTAR RUJUKAN

Alahmadi, A., dkk. (2023). A New Code Based Signature Scheme for Blockchain Technology. *Mathematics, 11*(1177), 1-20. doi:10.3390/math11051177

Az-Zuhaili, W. (2016). *Tafsir Al-Munir Jilid 2.* Jakarta: Gema Insani.

Az-Zuhaili, W. (2016). *Tafsir Al-Munir Jilid 3.* Jakarta: Gema Insani.

Bai, J., & Chang, C.-C. (2016). A High Payload Steganographic Scheme for Compressed Images with Hamming Code. *International Journal of Network Security, 18*(6), 1122-1129.

Baldi, M., Bianchi, M., & Chiaraluce, F. (2016). Enhanced Public Key Security for the McEliece Cryptosystem. *Journal of Cryptology, 29*, 1-27. doi:10.1007/s00145-014-9187-8

Bashardoost, M., Rahim, M., & Hadipour, N. (2015). A Novel Zero-Watermarking Scheme For Text Document Authentication. *Jurnal Teknologi, 75*(4), 49-56. doi:10.11113/jt.v75.5066

Basri. (2016). Kriptografi Simetris dan Asimetris dalam Perspektif Keamanan Data dan Kompleksitas Komputasi. *Jurnal Ilmiah Ilmu Komputer, 2*(2), 16-23.

Dixit, A., & Dixit, R. (2017). A Review on Digital Image Watermarking Techniques. *I.J. Image, Graphics and Signal Processing, 4*, 56-66. doi:10.5815/ijigsp.2017.04.07

Dlamini, N., Mthethwa, S., & Barbour, G. (2018). Mitigating the Challenge of Hardcopy Document Forgery. *2018 Int. Conf. Adv. Big Data, Comput. and Data Commun. Syst. icABCD*. doi:10.1109/ICABCD.2018.8465401

Harianja, M. H. (2023). Analisa Fungsi Hash Untuk Mendeteksi Otentikasi File Video Menerapkan Metode N-Hash. *Management of Information System Journal, 2*(1), 7-13.

Haryadi, B. (2023). *Elektronika Digital.* Banyumas: Wawasan Ilmu.

Hillier, C., & Balyan, V. (2019). Error Detection and Correction On-Board Nanosatellites Using Hamming Codes. *Journal of Electrical and Computer Engineering*(1), 1-15. doi:10.1155/2019/3905094

Ilmiyah, N. F. (2018). Kajian Tentang Kriptosistem Mceliece Dalam Menghadapi Tantangan Komputer Kuantum Di Era Revolusi Industri 4.0. *Prosiding Seminar Nasional MIPA*, 216-226.

Jaameri, K. J. (2019). Code-based Cryptography. *School of Science*, 1-46.

Kementerian Agama. (2022a). *Qur'an Kemenag*. Retrieved from Lajnah Pentashihan Mushaf Al-Qur'an: https://quran.kemenag.go.id/quran/per-ayat/surah/23?from=8&to=8

Kementerian Agama. (2022b). *Qur'an Kemenag*. Diambil kembali dari Lajnah Pentashihan Mushaf Al-Qur'an: https://quran.kemenag.go.id/quran/per-ayat/surah/24?from=27&to=27

Kementerian Agama. (2022d). *Qur'an Kemenag*. Diambil kembali dari Lajnah Pentashihan Mushaf Al-Qur'an: https://quran.kemenag.go.id/quran/per-ayat/surah/4?from=58&to=58

Kementrian Agama. (2022c). *Qur'an Kemenag*. Diambil kembali dari Lajnah Pentashihan Mushaf Al-Qur'an: https://quran.kemenag.go.id/quran/per-ayat/surah/2?from=283&to=283

Khairiyah, N., dkk. (2024). Hash-Based Authentication Code Algorithm for Quick Response (QR) Code as Digital Signature. *International Journal of Emerging Technology and Engineering Education, 1*(1), 7-12. doi:10.24036/int.j.emerg.technol.eng.educ..v1i1.2

Kumar, M. (2022). Post-Quantum Cryptography Algorithm's Standardization and Performance Analysis. *Elsevier, 15*, 1-27. doi:10.1016/j.array.2022.100242

Kurniawan, F., Kusyanti, A., & Nurwarsito, H. (2017). Analisis dan Implementasi Algoritma SHA-1 dan SHA-3 pada Sistem Autentikasi Garuda Training Cost. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1*(9), 803-812.

Lorien, A., & Wellem, T. (2021). Implementasi Sistem Otentikasi Dokumen Berbasis Quick Response (QR) Code dan Digital Signature. *Rekayasa Sistem dan Teknologi Informasi (RESTI), 5*(4), 663-671. doi:10.29207/resti.v5i1.3316

Munir, R. (2019). *Kriptografi.* Bandung: Institut Teknologi Bandung.

Nurhayati, Kastari, & Fahrianto, F. (2022). End-To-End Encryption on the Instant Messaging Application Based Android using AES Cryptography Algorithm to a Text Message. *CITSM*. doi:10.1109/CITSM56380.2022.9935963

Prabowo, E. C., & Afrianto, I. (2017). Penerapan Digital Signature dan Kriptografi pada Otentikasi Sertifikat Tanah Digital. *Jurnal Ilmiah Komputer dan Informatika, 6*(2), 83-90. doi:10.34010/komputa.v6i2.2481

Puspitasari, Y. (2023). Etika Komunikasi Tentang Kejujuran dan Keadilan dalam Perspektif Al-Qur'an. *Tabayyun, 4*(1), 17-26. doi:10.61519/tby.v4i1.45

Rahim, I., dkk. (2023). Komparasi Fungsi Hash MD5 dan SHA256 dalam Keamanan Gambar dan Teks. *Ikraith-Informatika, 7*(2), 41-48.

Romine, C. H. (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. *Federal Information Processing Standards Publications*.

Santoso, M. H., dkk. (2019). Perbandingan Algoritma Kriptografi Hash MD5 dan SHA-1. *Semantika, 2*(1), 54-59.

Sari, M. P. (2021). Analisis Algoritma SHA-3 Keamanan pada Data Pribadi. *Tecnoscienza, 5*(2), 231-242.

Sawata, R., Kashiwagi, Y., & Takahashi, S. (2022). Improving Character Error Rate is Not Equal to Having Clean Speech: Speech Enhancement for ASR Systems with Black-Box Acoustic Models. *ICASSP* (hal. 1-456). Tokyo: IEEE. doi:10.1109/ICASSP43922.2022.9746398

Sembiring, M., & Simbolon, F. H. (2021). Perancangan Perangkat Lunak Pembelajaran Algoritma Hamming Code dalam Mencari Bit Error pada Komunikasi Data. *LOFIAN: Jurnal Teknik Informasi dan Komunikasi, 1*(1), 24-28.

Sintyaningrum, D. E., Muladi, & Ashar, M. (2022). Implementasi Digital Signature dan QR Code Pada Sertifikat Profesi Digital untuk Mengatasi Kasus Pemalsuan Sertifikat. *Teknologi Elektro dan Kejuruan (TEKNO), 32*(1), 185-194.

Sinurat, S., & Siagan, E. R. (2022). Learning Text Data Security in Documents Using McEliece's Algorithm. *Jurnal Infokum, 10*(5), 323-330.

Sitorus, N., Sinaga, J. S., & Samosir , S. L. (2024). Analisis Kinerja Algoritma Hash pada Keamanan Data: Perbandingan Antara SHA-256, SHA-3, dan Blake2. *Jurnal Quancom, 2*(2), 9-16. doi:10.62375/jqc.v2i2.432

Suantara, P. Y., Satwika, P., & Fredlina, K. Q. (2024). Perbandingan Kinerja Waktu Algoritma ECDSA, EdDSA, RSA, dan Implementasinya pada Sistem Multi-Signature Dokumen PDF. *Jatisi, 11*(1), 85-98.

Supriyanto. (2020). *Sistem Komputasi (C2) Kelas X.* Malang: PT Kuantum Buku Sejahtera.

Upadhyay, D. dkk., (2022). Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications. *IEE Access, 10*, 112472-112486. doi:10.1109/ACCESS.2022.3215778

Varela, N., Ospino, C., & Lezama, O. B. (2020). Methodology for Processing Time Series Using Machine Learning. *Procedia Comput. Sci., 175*, 659-664. doi:10.1016/j.procs.2020.07.096

Wellem, T., Nataliani, Y., & Iriani, A. (2022). Academic Document Authentication using Elliptic Curve Digital Signature Algorithm and QR Code. *International Journal on Informatics Visualization, 6*(3), 667-675. doi:10.30630/joiv.6.2.872

LAMPIRAN

Lampiran 1. Dokumen Asli *dokumen1.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 2. *Signing* Dokumen *dokumen1_sign_key1.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 3. *Signing* Dokumen dokumen1_sign_key2.pdf

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 4. *Signing* Dokumen *dokumen1_sign_hapus.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 5. *Signing* Dokumen *dokumen1_sign_ubah.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490655)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 6. *Signing* Dokumen *dokumen1_sign_tambah.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW dengan membayar Rp 10.000,-. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490655)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 7. *Verifying* Dokumen *dokumen1_verif_key1.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 8. *Verifying* Dokumen *dokumen1_verif_key2.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 9. *Verifying* Dokumen *dokumen1_verif_hapus.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490650)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 10. *Verifying* Dokumen *dokumen1_verif_ubah.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490655)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

Lampiran 11. *Verifying* Dokumen *dokumen1_verif_tambah.pdf*

Hallo Vadillatul Ni'ma Maulidya (*Graphic Designer*)

Setelah melihat pengalaman dan juga CV kamu, kami sangat terkesan dengan pengalaman yang kamu miliki dan dengan senang hati menyatakan kamu LOLOS ke tahap selanjutnya yaitu INTERVIEW dengan membayar Rp 10.000,-. Harap simak hal-hal berikut ini sebelum melanjutkan ke tahap selanjutnya:

1. Silakan lakukan pengisian jadwal Interview di sheet berikut dan lakukan konfirmasi kepada kontak berikut paling lambat pada Rabu, 20 November, pukul 12.00 WIB. Konfirmasi bersifat WAJIB bagi semua yg sudah mengisi jadwal.

   Link Pengisian Jadwal Interview : https://bit.ly/Interview_MaqdisAcademy
   Konfirmasi ke : Reeza (wa.me/6285862490655)

2. Silakan masuk ke group ini untuk info lebih lanjut
   https://chat.whatsapp.com/E5bUh6V4YoPFeXKs1VmWR4

Wassalamu'alaikum wr.wb
Human Resource Development
Yayasan Maqdis

Note : Dilarang menyebarluaskan dan/atau meneruskan pesan ini kepada pihak dan/atau siapapun tanpa izin

RIWAYAT HIDUP

Vadillatul Ni'ma Maulidya, biasa di sapa Vadil yang merupakan putri pertama dari pasangan bernama Jumi'ati dan Ahmad Naim. Lahir di Kota Batu pada tanggal 10 Mei 2003 dan tinggal di Jl. Cempaka, Pesanggrahan, Batu.

Penulis menempuh pendidikan mulai dari MI Bustanul Ulum di Kota Batu yang berhasil diselesaikan pada tahun 2015. Kemudian, penulis melanjutkan pendidikan menengah pertama di SMP Negeri 02 Batu dan berhasil lulus pada tahun 2018. Selanjutnya, penulis melanjutkan pendidikan menengah atas di SMA Negeri 3 Batu dan berhasil lulus pada tahun 2021. Pada tahun yang sama, penulis melanjutkan pendidikan di perguruan tinggi UIN Maulana Malik Ibrahim Malang dan memilih Program Studi Matematika.

Selama masa kuliahnya, penulis tidak hanya berfokus pada akademik semata, namun juga aktif berpartisipasi dalam berbagai kegiatan organisasi baik intra maupun ekstra kampus. Salah satunya yaitu menjadi Ketua Divisi Penerbitan dan Jurnalistik HMPS "Integral" Matematika, di mana penulis dapat mengembangkan bakat dan minatnya dalam jurnalistik dan desain grafis. Selain itu, dalam hal akademik penulis juga aktif dalam program asistensi laboratorium selama 3 semester, di mana penulis dapat belajar lebih banyak terkait pemrograman dengan mahasiswa lain untuk mengembangkan diri dan wawasan dalam dunia teknologi khususnya pemrograman komputer.

Demikianlah gambaran singkat mengenai perjalanan hidup penulis. Dengan komitmen yang kuat dan usaha yang gigih selama masa perkuliahan, penulis dapat mengembangkan potensi bakat dan minatnya serta berhasil menyelesaikan tugas akhir tanpa terkendala.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No.50 Dinoyo Malang Telp. / Fax. (0341)558933

BUKTI KONSULTASI SKRIPSI

Nama : VADILLATUL NI'MA MAULIDYA
NIM : 210601110059
Fakultas / Jurusan : Sains dan Teknologi / Matematika
Judul Skripsi : Implementasi Algoritma SHA-3 dan McEliece dengan Kode Hamming untuk Otentikasi Dokumen Berbasis *Digital Signature*
Pembimbing I : Muhammad Khudzaifah, M.Si.
Pembimbing II : Abdul Aziz, M.Si.

No.	Tanggal	Hal	Tanda Tangan
1.	13 Agustus 2024	Konsultasi Topik dan Data	1.
2.	24 September 2024	Konsultasi Bab I, II, dan III	2.
3.	10 Oktober 2024	Konsultasi Bab I, II, dan III	3.
4.	21 Oktober 2024	Konsultasi Bab I, II, dan III	4.
5.	7 November 2024	Konsultasi Bab I, II, dan III	5.
6.	11 November 2024	ACC Bab I, II, dan III	6.
7.	10 Oktober 2024	Konsultasi Kajian Agama Bab I dan II	7.
8.	29 Oktober 2024	ACC Kajian Agama Bab I dan II	8.
9.	11 November 2024	ACC Seminar Proposal	9.
10.	24 Februari 2025	Konsultasi Revisi Seminar Proposal	10.
11.	20 Maret 2025	Konsultasi Bab IV dan V	11.
12.	27 Maret 2025	Konsultasi Bab IV dan V	12.
13.	28 April 2025	Konsultasi Bab IV dan V	13.
14.	30 April 2025	Konsultasi Bab IV dan V	14.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No.50 Dinoyo Malang Telp. / Fax. (0341)558933

15.	30 April 2025	ACC Bab IV dan V	15.
16.	28 April 2025	Konsultasi Kajian Agama Bab IV	16.
17.	30 April 2025	ACC Kajian Agama Bab IV	17.
18.	30 April 2025	ACC Seminar Hasil	18.
19.	26 Mei 2025	Konsultasi Revisi Seminar Hasil	19.
20.	2 Juni 2025	ACC Sidang Skripsi	20.
21.	16 Juni 2025	ACC Keseluruhan	21.

Malang, 16 Juni 2025
Mengetahui,
Ketua Program Studi Matematika

Dr. Elly Susanti, M.Sc.
NIP. 19741129 200012 2 005