Jurnal Penelitian Rumpun Ilmu Teknik
Volume. 3, No. 4, November 2024
e-ISSN :2963-7813 dan p-ISSN : 2963-8178; Hal. 153-178
DOI: https://doi.org/10.55606/juprit.v3i4.4569
Available online at: Jurnal Penelitian Rumpun Ilmu Teknik

Implementasi Kriptografi Pengamanan Data Soal Ujian di Lingkungan Perguruan Tinggi Menggunakan Algoritma AES-256 dan SHA-256

[1]Iwan Setiadi, [2]Santi Widianti, [3]I Putu Prachanda Kayuan
[1,2,3]Ilmu Komputer, Sistem Informasi, Universitas Gunadarma, Depok, Indonesia

Jl. Margonda Raya No. 100, 16421, Depok, Indonesia
Email : iwangsa@staff.gunadarma.ac.id[1], santiw@staff.gunadarma.ac.id[2],
putukayuan06@gmail.com [3]

ABSTRACT : *This study discusses the implementation of cryptographic technology for securing exam questions in a college environment, focusing on the use of the AES-256 and SHA-256 algorithms. The purpose of this study is to protect exam questions from the threat of leaks and ensure the authenticity of the questions received by students. In the experimental method in the proposed scientific writing, exam questions will be encrypted using the AES-256 algorithm, while the SHA-256 algorithm is used to generate a unique hash value that maintains the integrity of the question data, where the hash itself is an alphanumeric code generated from the process of converting digital data through certain mathematical functions. The AES-256 and SHA-256 Cryptographic Algorithms are chosen as algorithms to secure exam questions, Oracle VM VirtualBox and Kali Linux as the types of software/operating systems chosen to run the AES-256 and SHA-256 cryptographic algorithms. expected with strong protection against the threat of data leaks and manipulation, and ensure that the exam questions received by students remain original and do not change and are maintained authenticity.*

Keywords: *Data, Exam questions, Cryptography, Algorithm.*

ABSTRAK : Penelitian ini membahas implementasi teknologi kriptografi untuk pengamanan soal ujian di lingkungan perguruan tinggi, dengan fokus pada penggunaan algoritma AES-256 dan SHA-256. Tujuan dari penelitian ini adalah untuk melindungi soal ujian dari ancaman kebocoran dan memastikan keaslian soal yang diterima oleh mahasiswa. Dalam metode experimental dalam penulisan ilmiah yang diusulkan, soal ujian akan dienkripsi menggunakan algoritma AES-256, sedangkan algoritma SHA-256 digunakan untuk menghasilkan nilai *hash* unik yang menjaga integritas data soal, yang mana *hash* itu sendiri merupakan sebuah kode alfanumerik yang dihasilkan dari proses mengonversi data digital melalui fungsi matematika tertentu. Algoritma Kriptografi AES-256 dan SHA-256 di pilih sebagai algoritma untuk mengamankan soal ujian, Oracle VM VirtualBox dan Kali linux sebagai Jenis perangkat lunak/sistem Operasi yang dipilih untuk menjalankan algortima kriptografi AES-256 dan SHA-256. diharapkan dengan perlindungan yang kuat terhadap ancaman kebocoran dan manipulasi data, serta memastikan bahwa soal ujian yang diterima mahasiswa tetap asli dan tidak mengalami perubahan dan terjaga keaslianya.

Kata kunci : Data, Soal ujian, Kriptografi, Algoritma.

1. PENDAHULUAN

Dalam era digital ini, keamanan data menjadi hal yang sangat penting, termasuk dalam lingkungan pendidikan seperti perguruan tinggi. Dalam dunia pendidikan, integritas dan kepercayaan adalah dua hal yang sangat penting. Salah satu cara untuk memastikan kedua hal tersebut adalah dengan menjaga keamanan soal ujian. Dalam beberapa tahun terakhir, teknologi telah memainkan peranan penting dalam membantu institusi pendidikan. Teknologi kriptografi, khususnya algoritma AES-256 dan SHA-256, dalam konteks keamanan soal ujian di lingkungan perguruan tinggi digunakan untuk mengamankan soal ujian dari ancaman seperti kebocoran dan manipulasi data sedangkan memastikan integritas soal ujian dengan menggunakan algoritma SHA-256 yang dapat mendeteksi perubahan sekecil apapun pada data.

Received: September 16, 2024; Revised: Oktober 21, 2024; Accepted: November 25 , 2024;
Online Available: Desember 02, 2024;

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, integritas data, autentikasi, dan non-repudiasi. Algoritma SHA-256 adalah salah satu teknik kriptografi yang paling umum digunakan. Algoritma ini menghasilkan *hash* unik dari data yang diberikan, dan perubahan sekecil apapun pada data tersebut akan menghasilkan *hash* yang sangat berbeda. Ini membuat SHA-256 sangat efektif untuk memastikan integritas data.

Dalam konteks soal ujian, bisa menggunakan SHA-256 untuk mengintegrasikan soal ujian untuk menjaga keorisinalannya. Dengan cara kita mengambil teks dari soal ujian menguncinya dengan AES-256 yang nantinya file yang dihasilkan akan berupa **.aes**, kemudian kita mengintegrasikanya mengubah soal tersebut menjadi *hash value* menggunakan SHA-256, guna menjaga keaslian file, apa bila file tersebut telah dirubah isinya maka, *hash vlaue* yang dihasilkan akan berbeda, dengan ini dapat disimpulkan bahwa file telah terinterupsi . *Hash* ini kemudian disimpan dan digunakan sebagai kode pencocokan soal ujian yang asli dengan soal yang akan di gunalan. Ketika soal ujian diperlu, teks soal ujian dapat diambil kembali, diubah kembali menjadi file semula, lalu dapat dibandingkan dengan *hash* yang telah disimpan sebelumnya. Jika kedua *hash* cocok, maka soal ujian tersebut tidak diubah sejak terakhir kali disimpan.

Penerapan teknik ini dalam lingkungan perguruan tinggi memiliki beberapa keuntungan. Pertama, ini membantu memastikan bahwa soal ujian yang diberikan kepada mahasiswa adalah soal yang sama dengan yang telah disiapkan oleh dosen. Kedua, ini juga membantu mencegah kebocoran soal, karena soal yang bocor akan memiliki *hash* yang berbeda dengan *hash* yang asli. Namun, penerapan teknik ini juga memiliki beberapa tantangan. Misalnya, kita perlu memastikan bahwa proses konversi data atau informasi, seperti teks atau file, menjadi string karakter dengan panjang tetap menggunakan fungsi *hash* tertentu, atau sering di sebut dengan hash*ing*, *hashing* dilakukan dengan benar dan *hash* yang dihasilkan disimpan dengan aman. Selain itu, kita juga perlu memastikan bahwa sistem yang digunakan untuk mengimplementasikan teknik ini aman dari serangan eksternal.

2. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah metode Experimental. Penelitian ini hanya sampai tahapan percobaan implementasi terhadap algoritam AES-256 yang telah di unduh dari internet dan web SHA-256 online tools, sebagai sebagai sarana yang di guanakan untuk melakukan enkripsi dan dekripsi terhadap soal ujian yang akan dilakukan uji coba. Pengamanan soal ujian menggunakan Kali linux. Data soal ujian yang diambil berasal dari

soal UTS (Ujian Tengah Semester) Universitas Gunadarma sebagai objek eksperimen. Data set yang di gunakan sebanyak satu data soal ujian yang akan dilakukan ujicoba.

Penelitian ini dimulai dengan tahapan Perencanaan, pada tahapan ini, penulis melakukan pencarian atau observasi terhadap perangkat lunak/sistem oprasi Oracel VM VirtualBox dan Kali Linux, untuk algoritma yang dipilih adalah AES -256 dan SHA-256. Tahap selanjutnya adalah tahap analisis, analisis dilakukan untuk mengidentifikasikan gambaran umum dan teori terhadap algoritma yang akan digunakan dan sarana-sarana apa saja yang akan di gunakan nantinya dalam proses penelitian. Pada tahap perancangan penulis melakukan perancangan terhadap struktur dan proses pengimplementasian prosesn enkripsi dan dekripsi algoritma. Terakhir adalah tahap pengujian dan analisis hasil, pada tahap ini dilakukan pengujian untuk mengevaluasi kefektivitasan terhadap algoritma dalam penerapan keamanan pada soal ujian secara keseluruhan.

3. HASIL DAN PEMBAHASAN

Gambaran Umum Aplikasi

Penelitian ini berfokus dalam pengimplementasi teknologi kriptografi untuk pengamanan soal ujian di lingkungan perguruan tinggi dengan memanfaatkan algoritma AES-256 dan SHA-256, menggunakan metode eksperimental di mana soal ujian dienkripsi dengan algoritma AES-256 untuk memastikan kerahasiaannya. Sementara itu, algoritma SHA-256 diterapkan untuk menghasilkan nilai *hash* unik yang menjaga integritas data soal, memastikan bahwa soal tetap asli dan tidak mengalami perubahan. Diharapkan penelitian ini dapat menciptakan sistem pengamanan soal ujian, yang tidak hanya melindungi dari kebocoran dan manipulasi, tetapi juga memastikan keaslian soal yang diterima oleh mahasiswa. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam pengembangan sistem pengamanan data di lingkungan perguruan tinggi.

Implementasi Algoritma SHA-256 untuk *Hashing*

Tahap ini fokus pada penerapan algoritma SHA-256 untuk menghasilkan nilai *hash* dari data soal ujian. *Hash* ini berfungsi sebagai tanda tangan digital yang unik untuk memastikan integritas data selama proses pengiriman dan penyimpanan.

Implementasi Algoritma AES-256 untuk Enkripsi

Selanjutnya, data soal ujian yang telah dipersiapkan akan dienkripsi menggunakan algoritma AES-256. Enkripsi ini bertujuan untuk melindungi kerahasiaan data, sehingga hanya pihak yang berwenang yang dapat mengakses informasi tersebut.

Penggabungan dan Penyimpanan Data Terenkripsi

Setelah data dienkripsi dan *hash value* dihasilkan, keduanya digabungkan dan disimpan di lokasi yang aman. Penyimpanan ini dapat dilakukan di server internal atau media penyimpanan yang telah ditentukan.

Pengiriman Data dan Proses Dekripsi

Data terenkripsi kemudian dikirimkan ke penerima melalui jalur komunikasi yang aman. Setelah diterima, data tersebut akan didekripsi oleh penerima menggunakan algoritma AES-256 untuk memperoleh kembali data asli.

Verifikasi Integritas Data

Tahap akhir adalah verifikasi integritas data, di mana data yang telah didekripsi dibandingkan dengan nilai *hash* yang dihasilkan pada tahap awal. Jika hasilnya sesuai, data dianggap tidak mengalami perubahan, dan proses dapat disimpulkan berhasil menjaga integritas dan keamanan data.

Analisis Hasil dan Evaluasi

Tahap ini melibatkan analisis hasil dari setiap langkah yang telah dilakukan untuk mengevaluasi efektivitas sistem yang diimplementasikan. Evaluasi ini mencakup pengujian keamanan, kecepatan, dan keandalan sistem dalam menjaga kerahasiaan serta integritas data soal ujian.

Analisis Kebutuhan Perangkat

Perangkat keras yang digunakan dalam penelitian ini adalah Acer Nitro 5 dengan spesifikasi sebagi berikut :

Processor : 12th Gen Intel(R) Core(TM) i5-12500H 3.10 GHz
RAM : 16,0 GB
ROM : 474 GB
GPU : Intel Core i5
Sistem Operasi : Windows 11 Home Single Language

Spesifikasi ini dipilih untuk menjamin bahwa perangkat keras memiliki kapasitas yang memadai untuk mendukung aplikasi desain dan pengujian yang dibutuhkan selama proses penelitian.

Perangkat lunak yang digunakan dalam penelitian ini meluputi :

1. Oracle VM VirtualBOx : virtualisasi yang memungkinkan pengguna untuk menjalankan beberapa sistem operasi secara bersamaan pada satu komputer fisik. Dengan VirtualBox, kita dapat membuat dan mengelola mesin virtual di mana sistem operasi seperti Linux, Windows, atau lainnya dapat diinstal dan digunakan secara terpisah dari sistem operasi

utama. Ini sangat berguna untuk pengujian perangkat lunak, pengembangan, atau menjalankan lingkungan yang terisolasi tanpa mengubah konfigurasi sistem utama.

2. Kali Linux : merupakan distribusi Linux yang dirancang khusus untuk kebutuhan keamanan siber, pengujian penetrasi, dan forensik digital. Kali Linux dilengkapi dengan berbagai alat yang digunakan untuk pengujian keamanan, analisis jaringan, dan eksploitasi sistem, menjadikannya pilihan utama bagi profesional keamanan dan peneliti keamanan siber. Dengan dukungan terhadap berbagai teknik keamanan, Kali Linux sering digunakan dalam penelitian kriptografi dan pengamanan data.
3. Ekstensi Algoritma AES-256 : memungkinkan peningkatan keamanan melalui penggunaan kunci enkripsi yang lebih panjang dan struktur blok yang lebih kompleks. Hal ini memberikan perlindungan yang lebih kuat terhadap serangan kriptografi, seperti serangan brute force dan analisis diferensial, serta meningkatkan integritas dan kerahasiaan data yang dienkripsi.
4. Online Tools SHA-256: merupakan alat yang biasanya digunakan untuk memverifikasi integritas data dengan menghasilkan hash yang unik untuk setiap input, sehingga setiap perubahan pada data akan menghasilkan hash yang berbeda. Alat ini nantinya akan di akses melalui website online yang digunakan untuk melakukan autentikasi data soal ujian.

Cara keja AES-256

Sebagai contoh pengerjaan akan digunkan ” GunadarmaKampusD” sebgaia Plain Text dan "KampusDGunadarma" sebgai Key atau Kunci. Semua kata dan anggka harus di ubah ke dalam bilangan Hexadesimal. Ini di karenakan satu hexadesimal sama dengan emapat bit dan dua hexadesimal sama dengan delapan bit, yang mana 16 AES apabila dikalikan dengan delapan sama dengan 128bit, makan untuk 192bit 28 AES dan 256bit sama dengan 32 AES. Tahap pertama adalah melakukan persiapan plain text untuk di ubah ke dalam tabel, tabel ini nantinya akan di isi dengan bilangan hexsadesimal. Dan demi kemudahan pengerjaan cara kerja AES di sini penulis akan menggunakan AES 128 bit yang melakukan round sebanyak 10 kali.

Key Schedule

Plain text ”GunadarmaKampusD”

Maka akan di dapat : 47 75 6e 61 64 61 72 6d 61 4b 61 6d 70 75 73 44

47	64	61	70
75	61	4B	75
6E	72	61	73
61	6D	6D	44

Gambar 1. *Plain Text*

Chipper text "KampusDGunadarma"

Maka akan di dapat Chiper key : 4b 61 6d 70 75 73 44 47 75 6e 61 64 61 72 6d 61

4b	75	75	61
61	73	6e	72
6d	44	61	6d
70	47	64	62

Gambar 2. *Chipper text*

Key Schedule atau Penjadwalan kunci AES (Advanced Encryption Standard) adalah proses yang digunakan untuk menghasilkan serangkaian kunci putaran dari kunci awal. Kunci-kunci putaran ini kemudian digunakan dalam setiap putaran proses enkripsi dan dekripsi AES. Berikut adalah gambaran singkatnya: Setiap putaran kunci diturunkan dari kunci putaran sebelumnya menggunakan kombinasi operasi, termasuk:

- RotWord : Melakukan permutasi siklik pada kata.
- *SubBytes* : Menerapkan S-box AES ke setiap bit dari kata.
- Rcon : Menambahkan konstanta putaran ke kata

Untuk mencari w_i atau kolom yang bukan kelipatan empat, dengan cara melakukan operasi XOR dengan kolom sebelumnya atau w_{i-1} dengan kolom ke empat sebelumnya w_{i-4}

w_{i-4}			w_{i-1}		w_i			
4b	75	75	61					
61	73	6e	72					
6d	44	61	6d					
70	47	64	62					

Gambar 3. Persiapan Tabel Pemrosesan

Rotword

Fungsi ini digunakan untuk mengubah urutan bit dalam sebuah kata (word) yang terdiri dari 4 bit. Di sini RotWord akan selalu di lakukan pada kolom ke-empat.

61	RotWord =	72
72		6d
6d		62
62		61

Gambar 4. Hasil Dari Pemrosesn Rotword

SubBytes

Hasil RotWord akan dilakukan operasi *SubBytes* yaitu mensubtitusikan setiap kolom dengan dengan tabel S-box. S-box (Substitution box) itu sendiri merupakan tabel substitusi yang digunakan untuk menggantikan setiap bit input dengan bit lain. S-box ini dirancang untuk memberikan keamanan kriptografi dengan memperkenalkan non-linearitas ke dalam proses enkripsi dan dekripsi.Proses ini memastikan bahwa setiap bit output sangat berbeda dari bit inputnya, sehingga memperkuat keamanan enkripsi dengan membuat pola data lebih sulit untuk dianalisis dan dipecahkan oleh penyerang. Dapat di lihat pada gambar di bawah ini merupakan tabel S-Box.

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	la	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	46	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	6	24	5c	c2	de	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1b	9e
	e	e1	f8	98	11	69	d3	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar 5. Tabel S-Box

72	SubBytes =	a3
6d		3c
62		aa
61		ef

Gambar 6. Hasil Subtitusi S-Box

Rcon

Seperti yang sudah di jelaskan Rcon Ini merupakan matrix konstan yang nilainya digunakan dalam proses ekspansi kunci untuk meningkatkan keamanan. Setiap putaran dalam ekspansi kunci AES menggunakan nilai Rcon yang berbeda.

Rcon Matrix									
01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Gambar 7. Tabel Rcon

4b		a3		01		e9
61	⊕	3c	⊕	00	=	5d
6d		aa		00		c7
70		ef		00		9f

Gambar 8. Lanjutan Hasil Dari **Gambar 3. 6**

w_{i-4}			w_{i-1}		w_i			
4b	75	75	61		e9			
61	73	6e	72		5d			
6d	44	61	6d		c7			
70	47	64	62		9f			

Gambar 9. Hasil w_i Pemrosesan dari **GambaR 3.8**

Operasi Rcon ini hanya berlaku untuk Kolom kelipatan 4 saja, untuk mengisi 32 bit sisanya kolom yang kosong kita akan melakukan operasi XOR dengan posisi sebelumnya atau w_{i-1} dengan empat posisi sebelumnya w_{i-4}. Dengan menggunakan operasi sebagai berikut

75		e9
73	⊕	5d
44		c7
47		9f

Gambar 10. Proses *XOR Rcon*

Maka akan di dapat hasil sebagai berikut

w_{i-4}				w_{i-1}	w_i		
4b	75	75	61	e9	9c		
61	73	6e	72	5d	2e		
6d	44	61	6d	c7	83		
70	47	64	62	9f	d8		

Gambar 11. Hasil dari Proses Rcon Pertama

Operasi ini akan terus di lakukan sampai kolom yang kosong terisi semua, maka akan di dapat Roun key yang pertama, langkah ini akan di lakukan samapai Mendapatkan Round key yang ke sepuluh

			w_{i-4}				w_{i-1}	w_i
4b	75	75	61		e9	9c	e9	88
61	73	6e	72		5d	2e	40	32
6d	44	61	6d		c7	83	e2	8f
70	47	64	62		9f	d8	bc	de

Gambar 12. Hasil Akhir Dari Kelanjutan Proses Rcon

Setelah dilakukan operasi pencarian Round key maka didapat Round key yang ke-10

37	5a	56	46
cc	a4	a7	f5
e1	d2	ee	25
87	1b	47	88

Gambar 13. Round key Ke-10

Initial Round

Pada algoritma *Advanced Encryption Standard* (AES), initial round atau putaran awal adalah langkah pertama dalam proses enkripsi. Pada tahap ini, blok plaintext yang akan dienkripsi pertama kali mengalami operasi *AddRoundKey*, di mana kunci awal (*initial key*) ditambahkan ke blok *plain text* menggunakan operasi XOR. Langkah ini bertujuan untuk menggabungkan kunci dengan data asli sebelum memasuki putaran-putaran enkripsi berikutnya. Putaran awal ini sangat penting karena mempersiapkan data untuk transformasi lebih lanjut melalui serangkaian putaran yang melibatkan langkah-langkah seperti *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*.

47	64	61	70		4b	75	75	61
75	61	4b	75	⊕	61	73	6e	72
6e	72	61	73		6d	44	61	6d
61	6d	6d	44		70	47	64	62

Gambar 14. Hasil *Initial Round*

Untuk mempermudah melakukan operasi XOR, bilangan hexadesimal dari plain text dan chipper key harus di ubah ke dalam bilangan biner.

Maka hasil dari *initial round* adalah

0e	11	14	11
14	12	25	07
03	36	00	1e
11	2a	00	26

Gambar 15. Hasil dari *Initial Round*

SubBytes

Pada tahap ini hasil darioperasi initial round akan dilakukan subtitusi dengan tabel *S-Box* yang mana Tabel substitusi yang digunakan berfungsi untuk mengganti setiap bit input dengan bit lain. *S-Box* ini didesain untuk meningkatkan keamanan kriptografi dengan memperkenalkan elemen non-linear ke dalam proses enkripsi dan dekripsi.

ab	82	fa	82
fa	c9	3f	c5
7b	05.	63	72
82	e5	63	f7

Gambar 16. Hasil Operasi *SuBytes* Dengan *Tabel* S-Box

ShiftRows

Langkah ini bekerja pada setiap baris pada tabel. Baris pertama tidak diubah, sedangkan tiga baris terakhir digeser ke kiri satu, dua, dan tiga kali, secara berurutan. Untuk menggeser ke kiri sekali, bit paling kiri dipindahkan ke kolom paling kanan, dan tiga bit yang tersisa digeser satu kolom ke kiri.

ab	82	fa	82			ab	82	fa	82
fa	c9	3f	c5	Digeser sebanyak 1 byte		c9	3f	c5	fa
7b	05	63	72	Digeser sebanyak 2 byte		63	72	7b	05
82	e5	63	f7	Digeser sebanyak 3 byte		f7	82	e5	63

Gambar 17. Hasil Operasi *SiftRows*

MixColumns

Tahap ini merupakan difusi dalam AES, yang membantu menyebarkan bit input ke seluruh blok data sehingga membuat pola data lebih sulit untuk dikenali oleh penyerang.Untuk melakukan operasi *Mix Columns* pada matriks AES, kita perlu mengalikan setiap kolom dari matriks state dengan matriks tetap *Mix Columns*:

02	03	01	01			ab	82	fa	82
01	02	03	02			c9	3f	c5	fa
01	01	02	03		.	63	72	7b	05.
03	01	01	02			f7	82	e5	63
Matrix Columns						Matrix State			

Gambar 18. Persiapan Proses Perkalian *Matrix Cloumns* Dengan *Matrix State*

02	03	01	01		ab		e2
01	02	03	02		c9		d9
01	01	02	03	.	63	=	b6
03	01	01	02		f7		f1

Gambar 19. Kolom Pertama Sebagai Contoh

Sebagai contoh kolom pertama akan di gunakan dalam pengitungan matrix ini. Setelah melakukan operasi tersebut untuk setiap kolom, maka akan di dapat hasil akhirnya adalah:

e2	91	11	2b
d9	3e	52	4c
b6	8d	6d	5a
f1	2c	3f	7e

Gambar 20 Hasil Dari Akhir Mix Columns

AddRounKey

Pada tahap ini melibatkan operasi XOR (exclusive OR) antara matriks state, yang berisi data yang sedang dienkripsi, dan matriks kunci (Round key) yang sudah di temukan dari proses Key schedule. Operasi XOR dilakukan bit per bit, menggabungkan setiap elemen dari matriks state dengan elemen yang sesuai dari round key. Matriks state adalah representasi dari blok data yang sedang dienkripsi, sedangkan round key adalah kunci yang berbeda untuk setiap putaran enkripsi. *AddRoundKey* dilakukan pada setiap putaran enkripsi AES, termasuk sebelum putaran pertama dan setelah putaran terakhir. Langkah ini memastikan bahwa data yang dienkripsi menjadi lebih aman dengan menggabungkan kunci enkripsi secara langsung ke dalam data, sehingga membuat pola data lebih sulit dikenali oleh penyerang.

e2	91	11	2b		37	5a	56	46
d9	3e	52	4c	⊕	cc	a4	a7	f5
b6	8d	6d	5a		e1	d2	ee	25
f1	2c	3f	7e		87	1b	47	88
					Round Key			

Gambar 21. Proses XOR *AddRoundKey*

Maka didapatkan hasil akhir adri operaasi keseluruhan adalah d5155776cb9a5f3747f583786db97ff6.

d5	cb	47	6d
15	9a	f5	b9
57	5f	83	7f
76	37	78	f6

Gambar 22. Hasil Akhir Pengerjaan AES-256

Cara Kerja Algoritma SHA-256

Pada tahap ini terdiri dari tiga langkah prapemrosesan: menyisipkan padding pada pesan, *M*, membagi pesan menjadi blok-blok pesan, dan mengatur nilai *hash* awal, $H^{(0)}$. Tujuan dari padding ini adalah untuk memastikan pesan yang telah diberi padding memiliki kelipatan 512 atau 1024 bit, tergantung pada algoritma yang digunakan. Padding dapat disisipkan sebelum komputasi *hash* dimulai pada pesan, atau pada saat lain selama komputasi *hash* sebelum memproses blok yang akan berisi padding.

apabila panjang pesan, *M* adalah l bits. Tambahkan “1” diakhir pesan, diikuti oleh $k$bit nol, dimana k adalah solusi non-negatif terkecil untuk persamaan $l + 1 + k = 448 \bmod 512$.

Kemudian tambhakanblok 64-bit yang sama dengan jumlah l yang di representasikan menggunakan biner. Sebagai contoh, pesan (8-bit ASCII) "**abc**" memiliki panjang 8x3=24, sehingga pesan tersebut di beri padding dengan satu bit kemudian 448-(24+)=423 bit nol, dan kemudian panjang pesan, menjadi pesan yang diberi padding 512-bit. Panjang pesan yang telah diberi padding sekarang harus merupakan kelipatan dari 512 bit.

Parsing Pesan

Proses di mana pesan input diolah untuk menghasilkan nilai *hash* 256-bit. Proses ini dimulai dengan mengkonversi pesan ke format biner dan menambahkannya dengan padding untuk memastikan panjang totalnya sesuai dengan kelipatan 512 bit. Padding mencakup penambahan bit '1', diikuti oleh bit '0', dan diakhiri dengan menyertakan panjang asli pesan dalam bentuk 64-bit. Dimana pesan dan padding-nya harus diuraikan menjadi N blok *m*-bit .Setelah padding, pesan dibagi menjadi blok-blok berukuran 512 bit, dan setiap blok diproses secara terpisah menggunakan algoritma SHA-256. Selama pemrosesan, blok-blok ini melalui berbagai transformasi matematis dan bitwise yang melibatkan fungsi *hash* dan variabel inisialisasi yang tetap. Hasil akhir dari pemrosesan seluruh blok adalah nilai *hash* 256-bit yang unik dan tetap untuk pesan tersebut. *Hash* ini berfungsi sebagai representasi ringkas dan aman dari pesan asli, memastikan integritas data dan keamanan kriptografis.

SHA-1, SHA-224 dan SHA-256

Untuk SHA-1, SHA-224 and SHA-256, pesan dan padding-nya diuraikan menjadi *N* block 512-bit (N=1), $M^{(1)}$, $M^{(2)}$,….., $M^{(N)}$. Di kareanakan block 512 dapat di representasikan sebagai 16 kata 32-bit (512/16=32), maka setiap 32 bit dari pesan ke- **i** dinotasikan $M_0^{(i)}$, 32 bit berikutnya dinotasikan sebagai $M_1^{(i)}$, danseterusnya hingga $M_{15}^{(i)}$.

SHA-256

Untuk SHA-256, nilai *hash* awal $H^{(0)}$, terdiri dari delapan kata 32-bit berikut, dalam format hex:

$$M_0^{(0)} = 6a09e667$$
$$M_1^{(0)} = bb67ae85$$
$$M_2^{(0)} = 3c6ef372$$
$$M_3^{(0)} = a54ff53a$$
$$M_4^{(0)} = 510e527f$$
$$M_5^{(0)} = 9b05688c$$
$$M_6^{(0)} = 1f83d9ab$$
$$M_7^{(0)} = 5be0cd19$$

Sebagai contoh

$\sqrt{2}$ = 1.414213562373095048801688 7…

Pecahan = 0.414213562373095048801688 7…

Binary = 01101010000010011110011001100111

Hex = 6a09e667

$M_0^{(0)}$ = 6a09e667

Dan seterusnya hingga $M_7^{(0)}$

Kata-kata ini diperoleh dengan mengambil tiga puluh dua bit pertama dari bagian pecahan akar kuadrat dari delapan bilangan prima pertama. Hal ini dilakukan ini untuk memastikan bahwa nilai-nilai tersebut acak sehingga tidak ada celah maupun back door dalam algoritma ini.

SHA-224 and Sha-256 Constants

SHA-224 dan SHA-256 menggunakan urutan yang sama dari 64 kata konstan 32-bit, $K_0^{\{256\}}$, $K_1^{\{256\}}$, $K_2^{\{256\}}$,….., $K_{63}^{\{256\}}$. Setiap kata mewakili 32 bit pertama dari bagian pecahan akar kubik dari 64 bilangan prima pertama. Dalam format hex, kata-kata konstan ini adalah (dari kiri ke kanan):

(428a2f98) (71374491) (b5c0fbcf) (e9b5dba5) (3956c25b) (59f111f1) (923f82a4) (ab1c5ed5) (d807aa98) (12835b01) (243185be) (550c7dc3) (72be5d74) (80deblfe) (9bdc06a7) (c19bf174) (e49b69c1) (efbe4786) (0fc19dc6) (240calcc) (2de92c6f) (4a7484aa) (5cb0a9dc) (76f988da) (983e5152) (a831c66d0) (b00327c8) (bf597fc7) (c6e00bf3) (d5a79147) (06ca6351) (14292967) (27b70a85) (2e1b2138) (4d2c6dfc) (53380d13) (650a7354) (766a0abb) (81c2c92e) (92722c85) (a2bfe8al) (a81a664b) (c24b8b70) (c76c51a3) (d192e819) (d6990624) (f40e3585) (106aa070) (19a4c116) (1e376c08) (2748774c) (34b0bcb5) (391c0cb3) (4ed8aa4a) (5b9cca4f) (682e6ff3) (748f82ee) (78a5636f) (84c87814) (8cc70208) (90befffa) (a4506ceb) (bef9a3f7) (c67178f2)

Komputasi SHA-256 Hash

Sebelum komputasi hash dimulai untuk masing-masing algoritma hash aman, dimana nilai hash awal, $H^{(0)}$ bergantung pada ukuran inti pesan merupakan hash yang menggunakan fungsi dan konstanta sebelumnya. Penambahan (+) dilakukan modulo 2^{32}. Setiap blok pesan $M^{(1)}$, $M^{(2)}$,….., $M^{(N)}$, diproses secara berurutan, menggunakan langkah-langkah berikut

Rumus Perispan Message schedule, $\{W_t\}$:

$$w_t = \begin{cases} M_1^{(1)} & 0 \leq t \leq 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \leq t \leq 63 \end{cases}$$

Untuk $W_0 - W_{63}$Mengguanakn Rumus :

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

Untuk mencari σ_0 dan σ_1

$\sigma_0^{\{256\}}(x) = \text{ROTR}^7 \oplus \text{ROTR}^{18}(x) \oplus SHR^3(x)$

$\sigma_1^{\{256\}}(x) = \text{ROTR}^{17} \oplus \text{ROTR}^{19}(x) \oplus SHR^{10}(x)$

Inisialisasi Variabel Kerja

Menginisialisasi kedelapan variabel kerja, **a, b, c, d, e, f, g** dan **h,** dengan $(i-1)^{St}$

hash value :

a = H_0 = 6a09e667 = 01101010000010011110011001100111

b = H_1 = bb67ae85 = 10111011011001111010111010000101

c = H_2 = 3c6ef372 = 00111100011011011111001101110010

d = H_3 = a54ff53a = 10100101010011111111010100111010

e = H_4 = 510e527f = 01010001000011100101001001111111

f = H_5 = 9b05688c = 10011011000001010110100010001100

g = H_6 = 1f83d9ab = 00011111100000111101100110101011

h = H_7 = 5be0cd19 = 01011011111000001100110100011001

Variabel Kerja

Untuk $\boldsymbol{T_0}$ Sampai dengan $\boldsymbol{T_{63}}$

$T_1 = h + \Sigma_1^{\{256\}}(e) + Ch(e, f, g) + K_t^{\{256\}} + W_t$

$T_1 = \Sigma_1$ rotasi kanan sebanyak 6, 11 dan, 25 diikuti dengan penambahan bitwise modulo 2^{32}

$T_2 = \Sigma_0^{\{256\}}(a) + Maj(a, b, c)$

$T_2 = \Sigma_0$rotasi kanan sebanyak 2, 13 and, 22 diikuti dengan penambahan bitwise modulo 2^{32}

$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

Setelah rotasi kanan (*right rotate*), langkah selanjutnya dalam konteks SHA-256 adalah penambahan *bit-wise modulo*. Ini berarti melakukan penambahan bit per bit dari dua nilai dengan menggunakan operasi *modulo* 2^{32}. Operasi ini biasanya dilakukan untuk menghasilkan nilai yang dapat diterima dalam bentuk 32-bit, sesuai dengan spesifikasi SHA-256.

Nilai ke i^{th} hash sementara :

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

Setelah mengulangi langkah satu hingga empat sebanyak *N* kali (yaitu, setelah memproses $M^{(N)}$), hasil nilai hash yang dihasilkan setelah proses hashing (*digest*) 256-bit dari message.

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

Cara Kerja Algoritma SHA-256 :

Universitas Guandarma akan di gunakan sebagi contoh simulasi data yang akan di lakukan hasing secara manual

Langkah pertama mempersiapkan message schedule, yang mana mengubah “Universitas Guandarma” menjadi binary maka akan di dapat 01010101 01101110 01101001 01110110 01100101 01110010 01110011 01101001 01110100 01100001 01110011 00100000 01000111 01110101 01100001 01101110 01100100 01100001 01110010 01101101 01100001, yang mana ada 168 bit kemudian di tambahkan 1 karena *N*=1, lalu tambahkan 0 sampai 448 bit kemudian sisa 96 bit terakhir merepresentasikan panjang bit sesungguhnya, dalam kasus ini terdapat 184 bit, di dapat hasil dari 184 di konfersikan ke biner adalah 10111000. Maka akan di dapat

$$W_0 = M_o = 01010101011011100110100101110110$$
$$W_1 = M_1 = 01100101011100100111001101101001$$
$$W_2 = M_2 = 01110100011000010111001100100000$$
$$W_3 = M_3 = 01000111011101010110000101101110$$
$$W_4 = M_4 = 01100100011000010111001001101101$$
$$W_5 = M_5 = 01100001100000000000000000000000$$
$$..$$
$$W_{13} = M_{13} = 00000000000000000000000000000000$$
$$W_{14} = M_{14} = 00000000000000000000000000000000$$
$$W_{15} = M_{15} = 00000000000000000000000010101000$$

$w_0, w_1, \ldots, w_{62}, w_{63}$ Yang mana W_t memiliki panjang 32 bit, 15 baris dari 32 bit pertama sama dengan $W_0 = M_o$

Untuk mencari $w_0, w_1, \ldots, w_{62}, w_{63}$ menggunakan rumus

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

Pertama kita harus mengetahui σ_0 dan σ_1, utnuk mengetahui σ_0 dan σ_1 kita dapat menggunakan rumus :

$$\sigma_0^{\{256\}}(x) = \text{ROTR}^7 \oplus \text{ROTR}^{18}(x) \oplus SHR^3(x)$$
$$\sigma_1^{\{256\}}(x) = \text{ROTR}^{17} \oplus \text{ROTR}^{19}(x) \oplus SHR^{10}(x)$$

Sebagai contoh 32 bit pertama akan di gunakan untuk mendemonstrasikan bagai mana cara rumus tersebut :

X = 01010101 01101110 01101001 01110110

Untuk σ_0

ROTR^7 = 10101010 10110110 10110110 10101110

ROTR^{18} = 01101110 10110110 01010101 01010110

SHR^3 = 00001101 11010110 01010101 01011100

Maka hasil σ_0 setelah di lakukan penambahan bit wise modulo 2 adalah

01000010101111101111001111010111

Untuk σ_1

ROTR^{17} = 01101001011101010101101111011001

ROTR^{19} = 10101110110010110101011001011100

SHR^{10} = 00000000000101010101101110011010

Maka hasil σ_1 setelah di lakukan penambahan bit wise modulo 2 adalah 01101001011101010101101111011001

Maka untuk mencari W_{16}

$W_{16} = \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0$

W_{14} = 00000000 000000000 00000000 0000000

ROTR^{17} = 00000000 000000000 00000000 0000000

ROTR^{19} = 00000000 000000000 00000000 0000000

SHR^{10} = 00000000 000000000 00000000 0000000

σ_1 = 00000000 000000000 00000000 0000000

W_1 = 01100101 01110010 01110011 01101001

ROTR^{7} = 11010010 11001010 11100100 11100110

ROTR^{18} = 10011100 11011010 01011001 01011100

SHR^{3} = 00001100 10101110 01001110 01101101

σ_0 = 01000010 10111110 11110011 11010111

W_0 = 01010101 01101110 01101001 01110110

σ_0 = 01000010 10111110 11110011 11010111 +

W_9 = 00000000 000000000 00000000 0000000 +

σ_1 = 00000000 000000000 00000000 0000000 +

W_{16} = 01101100 11111010 11100000 01110111

Di dapat kan W_{16} = 01101100 11111010 11100000 01110111

Dan seterusnya sampai W63, dan akan didapat hasil dari $W_{16} - W_{63}$ adalah sebagai berikut :

Wt			
W0	01010101011011100110100101110110	W32	01110111101111011010110000110001
W1	01100101011100100111001101101001	W33	10000110000010010101111100010101
W2	01110100011000010111001100100000	W34	01111100000100001110101000110011
W3	01000111011101010110000101101110	W35	00000110101111111101001010110101
W4	01100100011000010111001001101101	W36	01111101100001000101110110010000
W5	01100001100000000000000000000000	W37	11100000100101010110011011110111
W6	00000000000000000000000000000000	W38	00111000010000010100011101000110
W7	00000000000000000000000000000000	W39	00111010011000111100000011000111
W8	00000000000000000000000000000000	W40	00101110111101001000111100110110
W9	00000000000000000000000000000000	W41	11110110001111111000101010100101
W10	00000000000000000000000000000000	W42	01011000111101001100100110000101
W11	00000000000000000000000000000000	W43	11000000000101010010010010001010
W12	00000000000000000000000000000000	W44	00101110101111111011001111111111
W13	00000000000000000000000000000000	W45	11110111000000100010011110111111
W14	00000000000000000000000000000000	W46	11111101110111111011001110000010
W15	00000000000000000000000000000000	W47	01000011100010110100010011010010
W16	10011000001011010101110101001101	W48	00001001100100111110011001101100
W17	01111000011000000100101000000	W49	11111101110110001001011110010101
W18	00000101110001101011111010010110	W50	01101010100101111011010000011011
W19	00010000100101001110001001000100	W51	11000110101101100101101010001110
W20	11111001111011010011111011000001	W52	01001011100010011000000101111001
W21	01001110111011101010111101100000	W53	10011111010001010101010100001010
W22	00111000100001101011100100101100	W54	11100101010101110110101111101110
W23	00011010011111001111001001001110	W55	10001110010010100110011110010111
W24	00000100000111100010000000000000	W56	00111101000001000001011001110100
W25	11101101001011110000111111100011	W57	00010011110000000000100111010011
W26	11100100100101011110011101001000	W58	11101110010001110011101100010111
W27	01100000001000111111111110110010	W59	00101000111010110000010001010100
W28	10011110011000101111101100000001	W60	00110001101110010110000110100110
W29	10111000101111100010111000010110	W61	10110101101001010011010110110110
W30	10001101011011100101011010100101	W62	10100110110110001011101010111101
W31	10110110011011000101110010000011	W63	11011101100010101001100111010110

Gambar 23. hasil dari $W_{16} - W_{63}$

Working Variabel

Untuk mencari Workining Variabel, menggunakan rumus yang telah ada Variabel Kerja sebagai berikut :

$$T_1 = h + \Sigma_1^{\{256\}}(e) + Ch(e, f, g) + K_t^{\{256\}} + W_t$$

$$T_2 = \Sigma_0^{\{256\}}(a) + Maj(a, b, c)$$

Untuk $\boldsymbol{T_0}$ Sampai dengan $\boldsymbol{T_{63}}$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

Dan akan di dapat hasil dari *working variable*/variable kerja adalah :

$$h = 11110000111101010111111101001111$$

$$g = 10101001011100011000011010111000$$

$$f = 00110000101000011101110110110100$$

$$e = 00110000101000011101110110110100$$

$$d = 11110000001111111000110011010111$$

$$c = 00001010000111101110010010111100$$

$$b = 10111101001000111111101011100110$$

$$a = 10101000000101111100010111100110$$

Menghitung Nilai Ke, Antara i^{th} dan $H^{(i)}$

Menghitung nilai ke antara i^{th} dan $H^{(i)}$ Lalu di ubah ke hexadesimal

$H_0^{(i)} = 00001110000100011110110010100101$ = 0e 11 ec a5

$H_1^{(i)} = 01100011011111110111010001101011$ = 63 7f 74 6b

$H_2^{(i)} = 11111001100100101110111001011000$ = f9 92 ee 58

$H_3^{(i)} = 10101111011011101101100111110110$ = af 6e d9 f6

$H_4^{(i)} = 01100011000101101101010101000111$ = 63 16 d5 47

$H_5^{(i)} = 01010011101100011111101000100010$ = 53 b1 fa 22

$H_6^{(i)} = 01010000001001011011011101011111$ = 50 25 b7 5f

$H_7^{(i)} = 00000101010100100101001111010001$ = 05 52 53 d1

Setelah mengulangi langkah satu hingga empat sebanyak *N* kali (yaitu, setelah memproses $M^{(N)}$), hasil nilai hash yang dihasilkan setelah proses hashing (digest) 256-bit dari message.

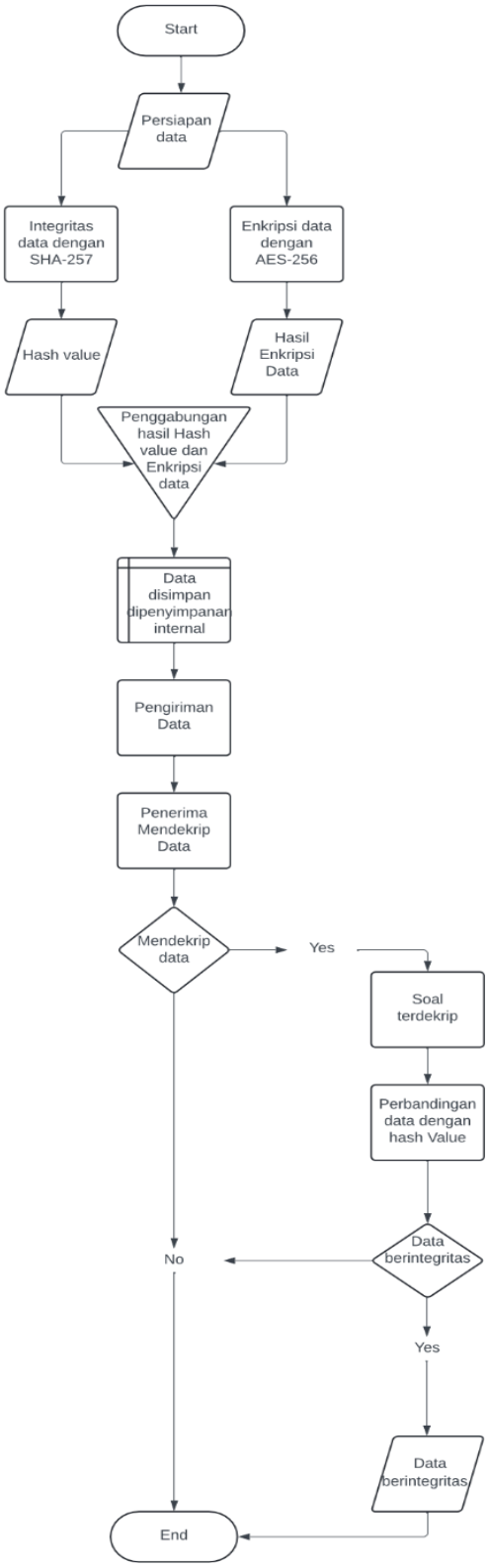
$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

Hash Value

Maka akan di dapat hasil dari proses hash Universitas Gunadarma :

0e11eca5637f746bf992ee58af6ed9f66316d54753b1fa225025b75f055253d1

Flowchart Alur Implementasi



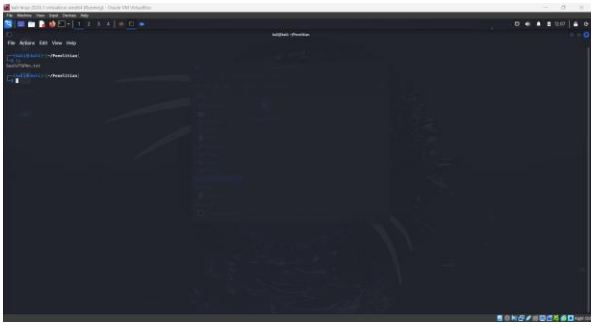
Gambar 24. Flowchart Alur implementasi

Flowchart ini menunjukkan alur dari proses enkripsi soal ujian samapai dengan proses dekripsi data soal ujian yang sistematis dalam pengamanan soal ujian menggunakan algoritma kriptografi, mulai dari persiapan data hingga verifikasi keaslian data setelah pengiriman. Berdasarkan flowchart yang dapat kita lihat pada gambar 24, berikut adalah penjelasan dari alur proses tersebut:

1. Start: Proses dimulai dengan langkah pertama, yaitu membuka Kali linux dan mempersiapan data yang akan diamankan. Data soal ujian yang digunakan sebagai Data uji coba berasal dari soal UTS Kewarganegaraan.

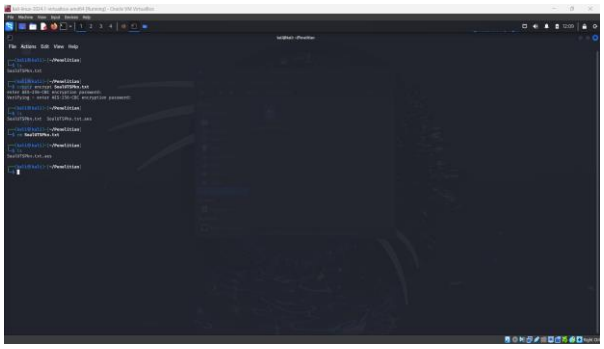
Gambar 25. Persiapan Data Soal Ujian.

2. Persiapan Data: Data yang akan dilakukan proses enkripsi dan diintegrasikan dipersiapkan terlebih dahulu. Kemudian data tersebut peneliti ubah ke dalam bentuk **.txt**, nantinya agar memepermudah dalam melakuakn integrasi soal ujian dengan menggunakan algoritma SHA-256



Gambar 26. File .dox Di Ubah Ke File .txt.

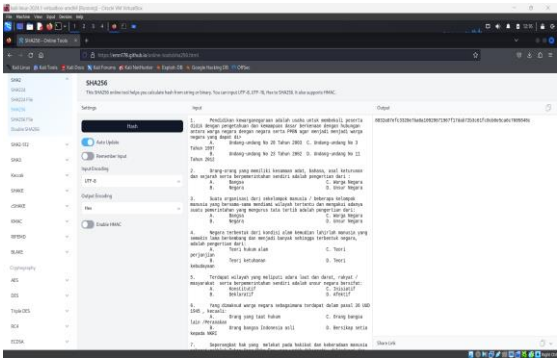
3. Enkripsi Data dengan AES-256: Data akan dienkripsi menggunakan algoritma AES-256 untuk menjaga kerahasiaan dan keamanannya. Pada tahap ini “KampusD” digunakan sebagai password dari file yang akan di enkripsi.



Gambar 27. Proses Enkripsi Algoritma AES-256

Dapat di lihat soal ujian telah berhasil di lakukan enkripsi, kemudain dilakuakn penghapusan terhadap file orisinalnya demi mencegah kebocoran data. Apabila file **.aes** tersebut dicoba untuk dibuka maka file tersebut tidak akan pernah bisa dibuka kecuali pihak yang bekepentinga mengetahui password file tersebut.

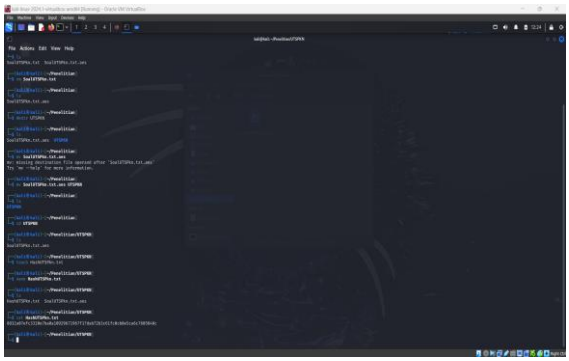
4. Integritas Data dengan SHA-256: Data yang telah dipersiapkan kemudian diuji integritasnya dengan menghasilkan nilai hash menggunakan algoritma SHA-256 online tools yang dapat di akses melalui website. Hash ini adalah representasi unik dari data asli, yang nantinya hasil output atau hash value akan di gunakan untuk perbandingan data hash untuk melakuak check integritas data.



Gambar 28. Hahing Menggunakan SHA-256 Online tools.

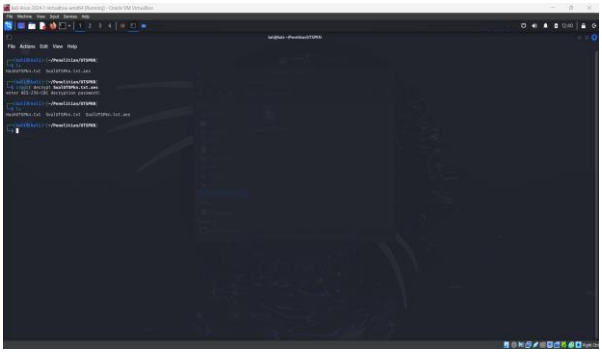
Dapat dilihat hasil dari hash terhadap hasil soal ujian tersebut adalah 8832a07efc3320e7ba0a10929671967f17dab72b3c61fc0cb0e5ca6c7605848c,hash value ini nantinya akan di simpan ke dalam file yang sama dengan SoalUTSPkn.txt.aes.

5. Penggabungan Hash Value dan Hasil Enkripsi Data: Setelah data terenkripsi dan nilai hash telah dihasilkan, keduanya dimasukan kedalam satu file yang sama untuk memastikan bahwa data dan hash value dienkripsi tidak terpisah, dan nantinya dapat dengan muda melakuakan verifikasi keasliannya.



Gambar 29. Proses Penyimpanan Data Enkrpsi Dan Hash.

6. Penyimpanan Data: Data yang telah dienkripsi dan dilengkapi dengan nilai hash kemudian disimpan, baik secara lokal atau di penyimpanan internal yang aman. Dapat dilihat pada gambar 3. Sudah dilakukan pembentukan direktori UTSPKN sebai tempat penyimpanan data hasil endkripsi AES-256 dan hash valuenya.
7. Pengiriman Data: Data yang telah disimpan kemudian dikirim ke penerima melalui jalur komunikasi yang ditentukan. Sebagai contoh menggunakan email atau prlatform media komunikasi lainya.
8. Penerima Menerima Data: kemudian pihak yang sudah di tentukan untuk menerima yang telah dikirim, harus mengakses data tersebut yang nantinya akan dilakukan pengecekan ulang data, apakah data yang dikirmkan oleh pengirim sama dengan data yang asli.
9. Mendekripsi Data: Penerima kemudian mendekripsi data menggunakan algoritma yang sama, AES-256, untuk mendapatkan kembali data asli. Apabila data tidak berhasil ter enkripsi proses akan terhenti sampai proses ini.

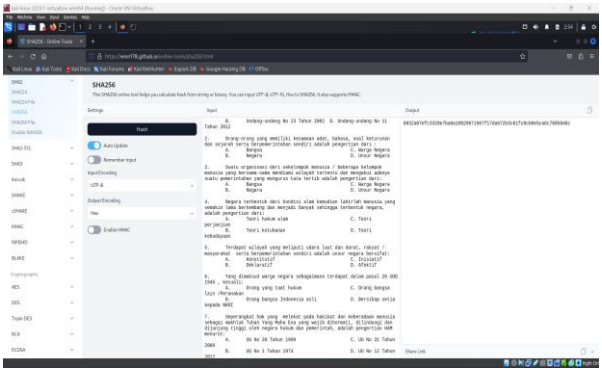


Gambar 30. Proses Dekripsi Soal Ujian

10. Soal Terdekripsi: Setelah proses dekripsi berhasil, soal ujian yang terenkripsi akan kembali menjadi bentuk aslinya, yaitu **.txt**.
11. Perbandingan Data dengan Hash Value: Data hasil dekripsi kemudian dibandingkan dengan nilai hash yang dihasilkan sebelumnya untuk memastikan tidak ada perubahan

selama proses pengiriman dan dekripsi. Sama seperti sebelumnya pada proses ini perbandingan hash value akan dilakukan menggunakan SHA-256 online tools, sebagai alat perbandingannya. Dengan cara melakukan copy terhadap keseluruhan isi file SoalUTSPkn.txt

12. Data Berintegritas: Jika data hasil dekripsi sesuai dengan nilai hash, maka data dianggap berintegritas (tidak berubah).



Gambar 31. Perbandingan Nilai Hash.

13. Data Tidak Berintegritas: Apabila nilai hash value yang di dapatkan berbeda dari nilai hash value yang telah di simpan maka Data tersebut telah terdistrupsi dan tidak original, dan proses akan kembali untuk pemeriksaan ulang atau pengiriman ulang dengan melakukan koordinasi ulang dengan pihak yang terkait.
14. End: Setelah seluruh proses berhasil dan integritas data terjaga, proses berakhir.

4. KESIMPULAN DAN SARAN

Kesimpulan

Algoritma AES-256 dan SHA-256 membuktikan kapabilitasnya dalam mengamankan soal ujian di perguruan tinggi. Dengan menggunakan kedua algoritma ini, soal ujian yang disimpan dan didistribusikan akan dijamin keasliannya, karena adanya hash value yang berfungsi sebagai acuan verifikasi. Fungsi hash ini bertindak sebagai kode verifikasi untuk memastikan keaslian data. Jika hash value yang diterima berbeda dari yang telah ditetapkan, maka data tersebut telah diubah atau tidak lagi orisinal.

Namun, terdapat beberapa kekurangan dalam proses implementasi ini. Proses hashing dan enkripsi masih dilakukan secara manual, yang memakan waktu meskipun tidak terlalu lama. Dalam praktiknya, terdapat banyak langkah yang harus dilakukan. Beberapa kendala lain yang sering muncul antara lain: algoritma AES-256 belum terinstal atau tidak termasuk dalam paket Kali Linux yang diunduh, serta adanya potensi kesalahan atau error dalam implementasi algoritma tersebut.

Saran

Penelitian ini masih jauh dari sempurna, untuk itu diharapkan penelitian ini menjadi satu batu inspirasi sebagai penelitian kedepannya, dan di harapkan penelitian dapat mengimplementasikannya secara nyata dan digunakan sebagai alat pengamanan soal ujian di lingkungan perguruan tinggi.

DAFTAR PUSTAKA

ENKRIPSI DAN DEKRIPSI DENGAN ALGORITMA AES 256 UNTUK SEMUA JENIS FILE Voni Yuniati (1) , Gani Indriyanta(2), Antonius Rachmat C(3). Diakses tanggal 27 Agustus 2024 dari https://media.neliti.com/media/publications/65826-ID-enkripsi-dan-dekripsi-dengan-algoritma-a.pdf

Cara Kerja AES. Di akses tanggal juli 2024 dari https://www.youtube.com/watch?v=61GRG4YloqU&t=412s

AES Animation. Di akses tanggal juli 2024 dari https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html

Difference Between AES and SHA256 by Diptendu Das. Diakses tanggal 27 Agustus 2024 dari https://diptendud.medium.com/difference-between-aes-and-sha256-706d6b2eb2ef

Advanced Encryption Standard (AES) by cosinesix 2023. Diakses tanggal 27 Agustus 2024 dari https://medium.com/@marketing_14184/advanced-encryption-standard-aes-dfb758ecbfa0

Pengamanan Arsip dengan Algoritma Enkripsi AES-256 untuk Web App E-Arsip Yayasan Universitas Islam Sumatera Utara Ibnu Hajar Fakultas Teknik, Teknik Informatika, Universitas Islam Sumatera Utara, Medan, Indonesia. Diakses tanggal 27 Agustus 2024 dari https://jurnal.ilmubersama.com/index.php/hello_world/article/download/13/27/382

PENERAPAN ALGORITMA KRIPTOGRAFI AES 256 UNTUK MENGAMANKAN DOKUMEN BERBASIS WEB PADA KELURAHAN BELENDUNG Irfan Kurnia Nurhareza, Siswanto Siswanto Fakultas Teknologi Informasi, Teknik Informatika, Universitas Budi Luhur, Jakarta, Indonesia. Diakses tanggal 27 Agustus 2024 dari https://senafti.budiluhur.ac.id/index.php/senafti/article/download/269/35 atau https://senafti.budiluhur.ac.id/index.php/senafti/article/view/269

A High-Performance Multimem SHA-256 Accelerator for Society 5.0 THI HONG TRAN, HOAI LUAN PHAM, and YASUHIKO NAKASHIMA. Diakses tanggal 27 Agustus 2024 dari https://www.researchgate.net/publication/349744176_A_High-Performance_Multimem_SHA-256_Accelerator_for_Society_50

Pengertian dan Fungsi SHA-256 dalam Melindungi Bitcoin dan Transaksi Digital. Diakses tanggal 27 Agustus 2024 dari https://support.bittime.com/hc/id/articles/8986445893903-Pengertian-dan-Fungsi-SHA-256-dalam-Melindungi-Bitcoin-dan-Transaksi-Digital

Analysis of SHA-512/224 and SHA-512/256 Christoph Dobraunig, Maria Eichlseder, and Florian Mendel Graz University of Technology, Austria maria.eichlseder@iaik.tugraz.at. Diakses tanggal 27 Agustus 2024 dari https://eprint.iacr.org/2016/374.pdf

National Security Agency . Diakses tanggal 27 Agustus 2024 dari https://en.wikipedia.org/wiki/National_Security_Agency

Quantum Computing Challenge . Diakses tanggal 27 Agustus 2024 dari https://medium.com/@kootie73/sha-256-from-origins-to-the-quantum-computing-challenge-57dd145ee690

SHA256: The Most Used Hash Function in Cryptocurrencies . Diakses tanggal 27 Agustus 2024 dari https://www.nervos.org/knowledge-base/SHA256_most_used_hash_function_(explainCKBot)

SHA-256 | FULL Step-by-Step Explanation (With Examples). Di akses tanggal juli 2024 dari https://www.youtube.com/watch?v=orIgy2MjqrA&t=2s

A Definitive Guide to Learn The SHA-256 (Secure Hash Algorithms) Lesson 10 of 64 By Baivab Kumar Jena. Diakses tanggal 27 Agustus 2024 dari https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

What Is the SHA-256 Algorithm & How It Works Last updated on August 5th, 2024 by Dionisie Gitlan. Diakses tanggal 27 Agustus 2024 dari https://www.ssldragon.com/blog/sha-256-algorithm/

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS). Diakses tanggal 28 Agustus 2024 dari https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

Let's dive in to the world of Virtualization by Tony Alosius, 2023**.** . Diakses tanggal 27 Agustus 2024 dari https://medium.com/@tony.aloysius.77/step-by-step-guide-to-installing-oracle-virtualbox-a-beginners-tutorial-eba18cfe6d2d

What is Kali Linux and How to Get Started? A Comprehensive Guide by The Intect, 2024. Diakses tanggal 27 https://medium.com/@info_82002/what-is-kali-linux-and-how-to-get-started-a-comprehensive-guide-ef97567375fe

Soal ujian yang di gunakan sebagai contoh data pengimplemantasian . Diakses tanggal 29 Agustus 2024 dari http://heliany.staff.gunadarma.ac.id/Downloads/files/38724/soal+uts+PKN.docx

AES-256 yang digunakan sebgai software tambahan, diunduh tanggal 27 agustus https://github.com/topics/crypter