IMPLEMENTASI ALGORITMA ADVANCED ENCRYPTION STANDARD (AES) UNTUK MENGENKRIPSI DATASTORE PADA APLIKASI BERBASIS ANDROID

Reski Mulud Muchamad [1], Asriyanik [2], Agung Pambudi [3]
[1,2,3] Program Studi Teknik Informatika Fakultas Sains dan Teknologi
Universitas Muhammadiyah Sukabumi
reski052@ummi.ac.id

ABSTRAK

Ponsel pintar merupakan perangkat yang sangat berguna untuk membantu aktivitas manusia. Ini terlihat dari tingkat penggunaan *smartphone* yang tinggi di seluruh dunia. Android adalah sistem operasi yang paling populer untuk perangkat ponsel pintar, sehingga banyak pengembang yang membuat aplikasi yang berjalan di perangkat Android. *DataStore* adalah media penyimpanan yang digunakan dalam aplikasi Android untuk menyimpan data pengguna. Namun, keamanan data pengguna masih merupakan perhatian utama bagi pengembang aplikasi, karena jika tidak dijaga dengan baik, data tersebut dapat disalahgunakan oleh pihak yang tidak berhak. Oleh karena itu, dalam penelitian ini akan diterapkan metode enkripsi menggunakan algoritma *Advanced Encryption Standard* (AES) pada *DataStore* untuk mengamankan data pengguna di aplikasi Android dengan mode *Cipher Block Chaining*. Algoritma AES memiliki keunggulan kecepatan dibandingkan dengan algoritma kriptografi lainnya. Hasil dari penelitian ini menunjukkan bahwa sistem yang dikembangkan memiliki tingkat risiko ancaman keamanan yang rendah, dan data yang disimpan dalam *DataStore* aman dari ancaman karena telah dienkripsi menggunakan algoritma AES yang dikembangkan. Selain itu, algoritma ini juga memiliki nilai *Avalanche Effect* yang baik, yaitu sebesar 64,68%, serta proses enkripsi dan dekripsinya lebih cepat dibandingkan penelitian sebelumnya.

Keyword : *Kriptografi, AES, Enkripsi, DataStore, Android*

1. PENDAHULUAN

Ponsel pintar (*smartphone*) adalah salah satu hasil dari kemajuan teknologi saat ini yang pesat. Tak dapat disangkal, keberadaan smartphone dapat membantu semua jenis pekerjaan yang dilakukan oleh manusia. Jumlah pengguna smartphone di seluruh dunia tercatat mencapai 5,3 miliar pada bulan Juli 2021 dengan persentase 67% dari populasi penduduk bumi (7,9 miliar)[1].

Android merupakan salah satu sistem operasi yang berjalan pada ponsel pintar. Android menjadi sistem operasi yang paling popular di dunia dengan lebih dari 2,5 miliar pengguna aktif di lebih dari 190 negara pada tahun 2022 [2]. Karena terdapat banyak sekali pengguna Android, maka banyak pula pengembang aplikasi Android, ini dibuktikan dengan banyaknya jumlah aplikasi yang telah rilis di *Google Play Store* dengan jumlah mencapai 2,7 juta aplikasi pada tahun 2020 [3].

Aplikasi yang tersedia di *platform* Android tentu sangat bervariasi, namun banyak dari aplikasi tersebut tentunya membutuhkan data pengguna, seperti nama, alamat email, kode token untuk berkomunikasi dengan API atau data lain yang serupa. Data yang diterima dari pengguna akan disimpan, baik secara lokal atau di internet.. Ada beberapa cara yang dapat digunakan untuk menyimpan data ke penyimpanan lokal, salah satunya adalah dengan menggunakan *DataStore* untuk menyimpan data pengguna dalam bentuk pasangan *key-value*. Selain *DataStore*, data dalam bentuk pasangan *key-value* juga dapat disimpan menggunakan *SharedPreferences*. Namun, *Android Developer* menyarankan untuk menggunakan *DataStore* atau melakukan migrasi ke *DataStore* jika sebelumnya masih menggunakan *SharedPreferences* [4].

Sebagai pengembang aplikasi Android, pasti sering terlibat dalam pengolahan data. Data yang diolah terkadang merupakan data yang sensitif dan berbahaya jika disalahgunakan. Fitur keamanan perlu diterapkan untuk mencegah ancaman yang disebabkan oleh kerentanan aplikasi terhadap serangan yang tidak diinginkan. Terdapat tiga prinsip utama dalam keamanan data, yaitu *confidentialy* (kerahasiaan), *integrity* (integritas), dan *availability* (ketersediaan), yang sering disingkat menjadi CIA [5] seperti yang ditunjukkan pada **Error! Reference source not found.** Dalam keamanan data, terdapat sebuah ancaman (*threats*) yang jika tidak diantisipasi akan menimbulkan risiko yang cukup besar. Seperti yang dijelaskan dalam penelitian yang dilakukan oleh [5] dijelaskan bahwa risiko terbesar terjadi pada reputasi dan kepercayaan pelanggan (*Reputation and Customer Confidence*) dengan hasil penilaian terhadap analisis risiko keamanan sebesar 12 dibandingkan dengan nilai *relative risk* sebesar 27. Contohnya, jika keamanan tidak diterapkan dengan baik dalam aplikasi Android, akan menyebabkan pengguna tidak percaya terhadap aplikasi dan berdampak buruk pada reputasi aplikasi yang dikembangkan.

Gambar 1. Prinsip utama keamanan data

Oleh karena itu, sangat penting untuk memperhatikan keamanan data pengguna dalam pengembangan aplikasi. Salah satu cara untuk mengamankan data yaitu menyembunyikan keaslian data dengan menerapkan algoritma kriptografi. Secara *default*, media penyimpanan yang tersedia dalam pengembangan aplikasi Android belum menerapkan enkripsi data, sehingga perlu metode tambahan untuk mengenkripsi data tersebut. Enkripsi dapat dilakukan dengan menggunakan pustaka yang disediakan langsung oleh *Android Developer* atau pustaka pihak ketiga.

Bagi *DataStore*, saat ini belum tersedia metode untuk mengenkripsi data yang disimpan di dalamnya. Berbeda dengan *SharedPreferences* yang telah disediakan metode untuk mengenkripsi data oleh *Android Developer*. *DataStore* merupakan metode atau media penyimpanan yang disarankan oleh *Android Developer* dibandingkan *SharedPreferences*, namun sayangnya saat ini masih belum tersedia fitur atau metode untuk mengenkripsi data di dalamnya. Isu ini juga disampaikan pada halaman *Issue Tracker Google* dengan jumlah lebih dari 170 dukungan (https://issuetracker.google.com/issues/167697691) dimana banyak yang menginginkan fitur untuk mengenkripsi *DataStore* [6]. Oleh karena itu, penulis melakukan penelitian ini dengan tujuan untuk menerapkan keamanan data pada *DataStore* menggunakan algoritma kriptografi.

Saat ini terdapat beragam jenis algoritma kriptografi yang tersedia, mulai dari tingkat kerumitan algoritma, jenis kunci yang digunakan, dan sebagainya. Salah satu algoritma yang dapat diterapkan adalah *Advanced Encryption Standard* (AES). Algoritma AES merupakan sebuah standar yang dikeluarkan oleh *National Institute of Standards and Technology* (NIST), di mana standar ini diberikan kepada algoritma *Rijndael* yang disetujui sebagai *Federal Information Processing Standard* (FISP) 197 pada tahun 2001 [7].

Algoritma *Advanced Encryption Standard* merupakan algoritma simetris, yang berarti proses enkripsi dan dekripsi menggunakan kunci atau *key* yang sama. Penelitian yang dilakukan oleh Nizirwan Anwar menunjukkan bahwa algoritma AES lebih cepat dibandingkan dengan algoritma RSA (asimetris), meskipun algoritma RSA lebih unggul dari segi keamanan karena menggunakan kunci atau key yang berbeda pada proses enkripsi dan dekripsi, dengan rata-rata kecepatan algoritma AES yaitu 236 kali lebih cepat dalam proses enkripsi dan 2,5 kali lebih cepat dalam proses dekripsi dibandingkan dengan RSA [8]. Sebenarnya algoritma AES sudah termasuk aman, seperti yang dijelaskan oleh Agung Prajuhana Putra melalui uji coba kemungkinan pemecahan kunci algoritma AES dengan panjang kunci 256 bit menggunakan metode *Brute Force* dengan asumsi kecepatan komputasi sebesar 106 kunci/detik, bahwa diperlukan waktu 3,8 x 10^{63} tahun untuk memecahkan kunci tersebut [9].

Penelitian terdahulu telah dilakukan untuk membandingkan algoritma AES dengan algoritma simetris lainnya. Dalam penelitian yang dilakukan oleh Elsa Elvira Awal, algoritma AES terbukti lebih cepat dibandingkan dengan algoritma *Twofish* pada proses enkripsi dan dekripsi dengan rata-rata algoritma AES membutuhkan waktu 3,94 detik untuk enkripsi dan 3,99 detik untuk dekripsi, sementara algoritma *Twofish* membutuhkan waktu 8,67 detik untuk enkripsi dan 9,02 detik untuk dekripsi [10]. Penelitian lain yang dilakukan oleh Donzilio Antonio Meko membandingkan algoritma DES, AES, IDEA, dan Blowfish. Hasilnya, algoritma AES terbukti lebih cepat dibandingkan dengan ketiga algoritma lainnya, dengan kecepatan proses enkripsi sebesar 48% dan kecepatan dekripsi sebesar 45% [11].

Berdasarkan penjelasan sebelumnya, dalam penelitian ini akan diterapkan algoritma Advanced Encryption Standard (AES) sebagai metode untuk mengenkripsi data pengguna yang disimpan di *DataStore* dalam aplikasi Android sebagai tindakan pengamanan. Untuk itu penulis melakukan penelitian ini yang bertujuan untuk mengamankan data yang disimpan ke dalam *DataStore* dengan mengimplementasikan algoritma *Advanced Encryption Standard* (AES).

2. TINJAUAN PUSTAKA

2.1. Keamanan Data

Keamanan data merupakan upaya yang dilakukan untuk menjaga rahasia serta memastikan bahwa tiga prinsip utama keamanan data terpenuhi [12]. Seperti yang telah disampaikan pada pembahasan latar belakang penelitian ini, prinsip utama keamanan tersebut yaitu *confidentialy* (kerahasiaan), *integrity* (integritas) dan *availability* (ketersediaan).

1. ***Confidentialy* (kerahasiaan) :** Prinsip kerahasiaan pada keamanan data melibatkan usaha melindungi data dari akses oleh orang yang tidak berhak mengaksesnya, seperti yang dijelaskan oleh Ika Yusnita Sari dalam bukunya tentang keamanan data dan informasi [13]. Prinsip kerahasiaan ini sangat berhubungan dengan privasi, di mana privasi adalah hal yang berkaitan dengan data yang dimiliki secara pribadi oleh seseorang. Terdapat tiga aspek dari privasi, yaitu *Privacy of Data about a Person* (privasi data seseorang), *Privacy of a Person's*

Persona (privasi mengenai pribadi seseorang) dan *Privacy of Person's Communication* (privasi komunikasi seseorang) [14]. Ketiga aspek tersebut harus dijaga kerahasiaannya, jika data tersebut digunakan oleh orang yang tidak berhak, orang tersebut dapat mengidentifikasi dirinya seolah-olah menjadi pemilik data yang sebenarnya. Hal ini dapat menimbulkan bahaya dan merugikan pemilik data asli.

2. ***Integrity* (integritas) :** Prinsip integritas yaitu menjamin bahwa data yang digunakan autentik, dikirim oleh orang yang benar, dan tidak diubah oleh orang yang tidak memiliki hak akses [13], [14].
3. ***Availability* (ketersediaan) :** Prinsip ketersediaan berkaitan dengan kemudahan akses data atau informasi yang diperlukan secara cepat dan tanpa penyembunyian atau gangguan dari pihak yang tidak memiliki hak akses [12]–[14].

2.2. Ancaman Keamanan Data

Ancaman keamanan merupakan peristiwa yang dapat merusak atau mengubah data atau sistem sehingga tidak lagi memenuhi prinsip keamanan yang seharusnya dijaga [5]. Ancaman keamanan dapat terjadi karena adanya celah atau kerentanan pada sistem, yang dapat diakses oleh kejahatan di dunia maya (*cyber crime*) untuk mengeksploitasi sistem tersebut dan mengakibatkan hilangnya prinsip keamanan [15]. Terdapat empat macam ancaman terhadap keamanan, yaitu gangguan (*interruption*), penyadapan (*interception*), modifikasi, dan pemalsuan (*fabrication*) [13], [14].

1. ***Interruption* (gangguan) :** Adalah ancaman yang terjadi pada perangkat sistem yang dapat menyebabkan kerusakan pada data, menyebabkan data yang dibutuhkan menjadi tidak tersedia. Ini dapat menghilangkan salah satu prinsip keamanan, yaitu ketersediaan (*availability*).
2. ***Interception* (pencekalan) :** Merupakan ancaman yang diakibatkan adanya orang atau pihak yang tidak berhak dalam mengakses data yang dimiliki seseorang.
3. ***Modification* (perubahan) :** Sesuai namanya, ancaman ini terjadi akibat terjadinya perubahan data yang dilakukan oleh seseorang yang tidak mempunyai hak akses terhadap data tersebut dan ancaman ini akan mengakibatkan hilangnya prinsip *integrity* (integritas) dalam prinsip keamanan.
4. ***Fabrication* (pemalsuan) :** Ancaman ini terjadi ketika terdapat seseorang yang tidak bertanggungjawab memalsukan data atau informasi seperti mengirim pesan-pesan palsu ke sistem. Sama seperti ancaman *modification*, ancaman ini juga menyebabkan hilangnya prinsip *integrity* (integritas).

Di dalam sistem perangkat ponsel dengan sistem operasi Android juga terdapat ancaman terhadap keamanan data yang ada di dalamnya. Karena Android merupakan sistem operasi yang memiliki lisensi sumber terbuka, maka dapat dikembangkan oleh orang lain atau komunitas, ini sangat baik karena dapat mempercepat proses pengembangan. Namun, di sisi lain hal ini menjadi suatu kelemahan karena semakin banyak orang yang memodifikasi maka akan semakin lemah keamanannya [12]. Selain itu terdapat ancaman dari luar sistem Android seperti serangan *malware*.

2.3. Kriptografi

Kriptografi adalah ilmu atau seni bagaimana melindungi data dengan mempelajari teknik matematika (Mukhtar, 2018). Kriptografi secara harfiah berasal dari kata Yunani "*crypto*" yang berarti "rahasia" dan "*graphia*" yang berarti "menulis". Terdapat beberapa istilah penting dalam kriptografi, dan menurut Intan Fitriani dalam penelitiannya menyebutkan empat istilah dalam kriptografi, yaitu:

1. ***Plain Text***, yaitu pesan asli yang belum mengalami proses enkripsi. Pesan ini mampu dibaca dan dimengerti maknanya.
2. ***Cipher Text***, yaitu sebuah pesan atau teks yang makna aslinya telah disembunyikan menggunakan metode kriptografi yaitu enkripsi. Makna yang terkandung di dalamnya sudah tidak dapat dipahami.
3. ***Cryptography Key*** merupakan sebuah kunci yang dimanfaatkan selama proses enkripsi dan dekripsi berlangsung.
4. ***Encryption and Decryption Algorithm*** Terdapat dua metode dalam kriptografi, yaitu proses untuk mengubah *plain text* ke dalam sebuah *cipher text*, dan proses untuk mengubah *cipher text* menjadi *plain text* yang bisa dibaca. Metode tersebut dinamakan enkripsi dan dekripsi.

Menurut penggunaan kunci, kriptografi terbagi menjadi dua, yaitu simetris dan asimetris. Kriptografi simetris menggunakan kunci yang sama ketika proses enkripsi dan dekripsi. Sedangkan kriptografi asimetris menggunakan kunci yang berbeda, pada proses enkripsi menggunakan kunci publik dan pada proses dekripsi menggunakan kunci privat. Pada penelitian ini penulis menggunakan algoritma kriptografi yang menggunakan kunci simetris.

2.4. Algoritma Kriptografi dan Metode yang Digunakan

Mengutip dari apa yang dijelaskan oleh Marsellus Oton Kadang dalam bukunya yang berjudul "Algoritma dan Pemrograman: Buku Bahan Ajar", algoritma adalah suatu teknik dalam menyusun langkah-langkah yang digunakan untuk menyelesaikan permasalahan yang disajikan dalam bentuk kalimat dengan jumlah kata yang terbatas namun tersusun secara sistematis dan juga logis [16]. Dalam buku tersebut juga dijelaskan bahwa

algoritma merupakan susunan langkah-langkah pasti yang bilamana diikuti akan menghasilkan data masukan yang ditransformasikan menjadi sebuah data keluaran yang berupa informasi.

1. **Kunci *Simetris,*** Metode ini menggunakan kunci yang sama pada saat proses enkripsi dan dekripsi. Jika menggunakan metode kriptografi simetris, maka harus menyimpan kunci dengan aman, karena jika diketahui oleh orang atau pihak yang tidak berhak maka akan memudahkan proses pembacaan data dan ini akan merugikan pemilik data sesungguhnya.
2. ***Blok Cipher*** adalah sebuah metode dalam algoritma kriptografi yang melakukan proses enkripsi dengan membagi *plain text* menjadi beberapa blok-blok bit, begitu juga kunci yang digunakan pun dibagi menjadi beberapa blok bit [17].
3. ***Advanced Encryption Standard*** merupakan sebuah standar untuk algoritma kriptografi yang dikeluarkan oleh *National Institute of Standard and Technology* (NIST) yang disetujui sebagai *Federal Information Processing Standard* (FISP) 197 pada tahun 2001, yang mana standar ini dibuat untuk menggantikan standar *Data Encryption Standard* (DES) yang dirasa sudah tidak aman karena menggunakan panjang kunci sebesar 56-bit [7], [18].

2.5. Android

Android adalah sebuah sistem operasi untuk perangkat ponsel yang dikembangkan oleh Google, sebuah perusahaan raksasa di bidang teknologi. Android merupakan sistem operasi yang berjalan di atas *kernel* Linux yang mendukung untuk berjalan pada perangkat layar sentuh; dan Android memiliki lisensi sumber terbuka yang membuat semua orang dapat menggunakan dan mengembangkan aplikasi Android secara gratis [19].

2.6. *DataStore*

Dilansir dari laman dokumentasi resmi *Android Developer*, *DataStore* merupakan sebuah solusi untuk penyimpanan data yang disimpan ke dalam penyimpanan lokal dalam bentuk pasangan *key-value* serta memanfaatkan Kotlin *coroutine* dan juga *Flow* [4]. *DataStore* termasuk ke dalam *Android Jetpack* yang merupakan kumpulan sebuah pustaka yang sangat bermanfaat untuk mengembangkan aplikasi Android. Di dalam laman dokumentasinya disampaikan pula bahwa *DataStore* disarankan untuk menggunakannya jika sebelumnya menggunakan *SharedPreferences*, ini karena *DataStore* menggunakan *Flow* yang mampu menjalankan proses penyimpanan data secara asinkron, konsisten dan transaksional, berbeda dengan *SharedPreferences* yang masih menjalankan prosesnya secara sinkron.

Dari uraian diatas juga diperkuat oleh penelitian yang sudah dilakukan oleh Agung Prajuhan Putra (2020) mengenai Implementasi Algoritma *Advance Encryption Standard* (AES) *Rijndael* pada Aplikasi Keamanan Data yang mana hasil dari penelitian tersebut yaitu implementasi algoritma *Advanced Encryption Standard* (AES) *Rijndael* dalam mengamankan data berupa berkas digital (*file*) yang disimpan ke dalam perangkat Android. Setelah itu di akhir penelitian dilakukan analisa dan pengujian dari performa serta keamanan algoritma AES dalam melakukan proses enkripsi dan dekripsi.

Selain itu, Intan Fitriani juga melakukan penelitian serupa mengenai implementasi algoritma *Advanced Encryption Standard* pada sistem Layanan SMS Desa. Penulis juga menjelaskan tentang beberapa pengujian untuk mengetahui efektifitas algoritma AES. Pengujian yang dilakukan antara lain adalah pengujian *CrackStation*, *Avalanche Effect*, dan uji kelayakan sistem yang dilakukan oleh ahli.

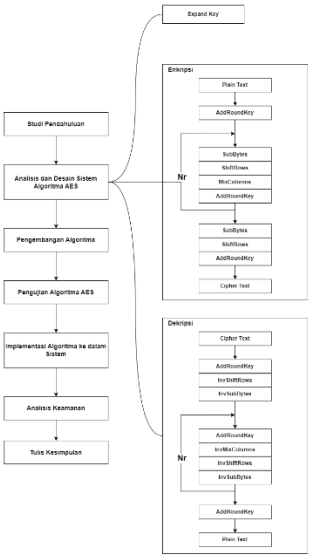
3. METODE PENELITIAN

Penelitian ini menggunakan metode yang terdapat pada algoritma *Advanced Encryption Standard* (AES) dengan menggunakan mode *Cipher Block Chaining* (CBC) untuk mengamankan datanya. Kemudian pada proses pencarian data juga dilakukan studi pendahuluan tentang *Advanced Encryption Standard* (AES) yang diperlukan untuk penelitian yang dilakukan.

Tahapan yang dilakukan pada penelitian ini yaitu:

1. **Studi Pendahuluan**: tahapan yang dilakukan yaitu dengan cara mencari informasi serta mengkaji penelitian terdahulu yang didapat dari sumber seperti jurnal dan buku terkait algoritma *Advanced Encryption Standard* (AES)
2. **Analisis Sistem Algoritma AES**: terdapat tiga tahapan dalam *Advanced Encryption Standard* (AES), yaitu pembangkit kunci, proses enkripsi dan proses dekripsi.
3. **Pengembangan Algotritma**: pada fase ini, akan dilakukan tahap pengembangan algoritma Advanced Encryption Standard (AES) dengan menggunakan bahasa pemrograman Kotlin.
4. **Pengujian**: tujuan pengujian adalah untuk memverifikasi kinerja enkripsi dan dekripsi, mengukur kecepatan proses dan memastikan keamanan sistem. Dalam rangka mengukur tingkat keamanan, metode *Avalanche Effect* digunakan sebagai acuan.
5. **Implementasi ke dalam Sistem**: pada tahap ini, implementasi atau penerapan algoritma akan dilakukan dalam aplikasi Android untuk mengenkripsi data yang disimpan dalam DataStore.
6. **Analisis Keamanan**: Tahap ini di-laksanakan dengan tujuan meng-identifikasi dan menemukan kerentanan pada aplikasi yang sudah dikembangkan. Metode yang digunakan

dalam analisis keamanan adalah dengan melakukan uji penetrasi atau penetration testing otomatis.



Gambar 2. Tahapan penelitian

4. HASIL DAN PEMBAHASAN

4.1. Kebutuhan Enkripsi Data Teks

Untuk memperjelas bagaimana proses enkripsi menggunakan algoritma *Advanced Encryption Standard* (AES) berjalan, maka akan dilakukan sebuah simulasi enkripsi dan dekripsi pada sebuah *plain text* menggunakan algoritma AES dengan panjang kunci 256-bit serta menggunakan mode *Cipher Block Chaining* (CBC) dengan beberapa ketentuan berikut:

- *Plain Text* : mengamankan data datastore dengan metode aes-256
- Kunci : Zr4u7x!A%D*G-KaPdSgVkXp2s5v8y/B*
- IV : WmZq4t7w!z%C*F-J

Sebelum melakukan tahapan yang lain, diperlukan untuk mengubah *plain text* ke dalam bentuk heksadesimal sesuai dengan tabel ASCII, lalu akan dibagi per blok dengan masing-masing panjang blok sebanyak 128-bit (16 karakter). Berikut adalah nilai *plain text* yang telah diubah ke dalam bentuk heksadesimal.

6d	61	6b	64
65	6d	61	61
6e	61	6e	74
67	6e	20	61

Blok 1

20	61	72	65
64	73	65	6e
61	74	20	67
74	6f	64	61

Blok 2

6e	74	20	2d
20	6f	61	32
6d	64	65	35
65	65	73	36

Blok 3

Gambar 3. Blok plain text

4.2. Proses Pembangkitan Kunci (*Expand Key*)

Seperti yang telah disampaikan sebelumnya, dibutuhkan kunci yang digunakan pada setiap ronde dalam proses enkripsi dan dekripsi. Berikut adalah simulasi dalam mengekspansi kunci, dan untuk tahap awal yaitu menyalin kunci ke dalam larik *W[]* seperti yang ditunjukkan di bawah.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	...	...	...	...	...	...	...	...
5a	37	25	2d	64	6b	73	79																
72	78	44	4b	53	58	35	2f																
34	21	2a	61	67	70	76	42																
75	41	47	50	56	32	38	2a																
Round 0				Round 1				Round 2				Round 3				Round n				Round 14			

Gambar 4. Menyalin blok kunci ke w_0

Tahap selanjutnya yaitu mencari nilai W_0 baru sampai semua *round key* didapatkan. Untuk mencari nilai W_0 berikutnya, terdapat beberapa tahapan yang harus dilalui, yaitu *RotWord*, *SubWord*, di XOR-kan dengan tabel *Rcon[i / Nk]* dan terakhir yaitu di XOR-kan dengan nilai W_{i-Nk}. Selanjutnya *word* hasil dari tahapan *RotWord* akan ditransformasikan pada tahap *SubWord* dengan tabel S-Box. Berikut adalah tabel S-Box yang digunakan.

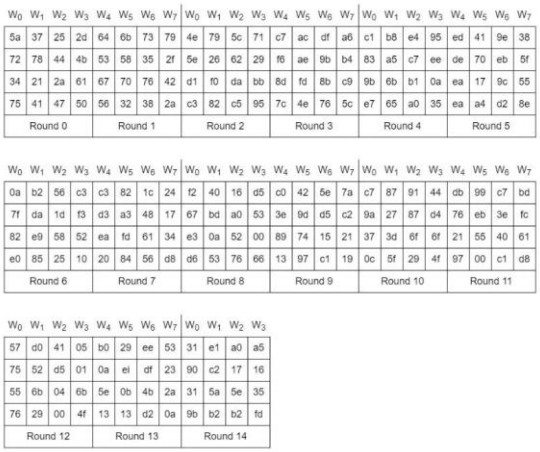
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	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	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	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	1d	9e
e	e1	f8	98	11	69	d9	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

Setelah melalui tahapan *SubWord*, hasil dari tahap tersebut akan di XOR-kan dengan tabel *Rcon[i / Nk]*. Tahap terakhir dalam mencari nilai W_0 selanjutnya yaitu dengan melakukan operasi XOR antara hasil *Rcon[]* dengan W_{i-Nk} (W_0 sebelumnya). Maka didapatkanlah hasil untuk nilai W_0 yang berikutnya setelah melalui berbagai tahapan.

Untuk mencari nilai W_1 sampai dengan W_7 berikutnya yaitu hanya dengan melakukan operasi XOR antara *word* sebelumnya (W_{i-1}) dengan W_{i-Nk}. Namun karena simulasi yang terdapat pada penelitian ini menggunakan kunci dengan panjang 256-bit, khusus saat mencari nilai W_4-nya akan melalui tahap *SubWord* terlebih dahulu [22]. Berikut adalah hasil dari nilai W_1 sampai W_7.

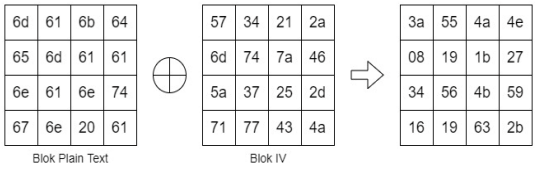
Setelah melalui tahapan yang sebelumnya, selanjutnya tahapan tersebut akan diulangi sampai seluruh nilai masing-masing *word* pada *round key* ditemukan. Berikut di bawah ini merupakan gambar hasil dari pencarian seluruh *word* untuk digunakan sebagai *round key* pada proses *Expand Key*.



Gambar 6. *Roundkey* yang digunakan untuk proses enkripsi dan dekripsi

4.3. Enkripsi Menggunakan AES

Proses enkripsi dilakukan pada setiap blok dan akan melalui setiap tahapan yang ada pada metode algoritma AES. Pada simulasi ini pertama kali akan melakukan enkripsi pada blok 1 *plain text*. Namun sebelum melakukan tahapan pada metode AES, terlebih dahulu blok *plain text* akan di XOR-kan dengan blok IV, ini dilakukan karena menggunakan mode CBC. Berikut adalah tahapan melakukan operasi XOR antara blok *plain text* dengan IV.



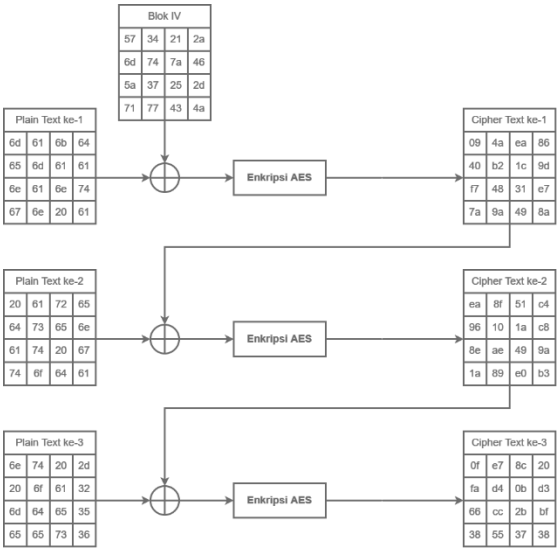
Gambar 7. Proses XOR blok plain text dengan IV

Tahapan selanjutnya yaitu *AddRoundKey* Ronde ke-0. Pada tahap ini akan melakukan operasi XOR antara *plain text* (yang telah di XOR dengan IV) dengan *round key* yang ke-0. Hasil dari tahapan ini akan disimpan ke dalam *state[]* (S_1) untuk diproses pada tahapan yang lain. Berikut adalah proses pada tahap *AddRoundKey* untuk ronde ke-0.

Tahapan selanjutnya yaitu *SubBytes*. Pada tahap *SubBytes* ini dilakukan proses transformasi *state[]* (S_1) sesuai dengan tabel S-Box yang sama dengan proses pembangkit kunci yang ditunjukkan pada **Gambar 5**. Hasil dari proses transformasi tersebut akan disimpan kembali ke dalam *state[]* (S_2). Tahap ini dilakukan terhadap setiap *bytes* pada *state[]*. Setelah ditransformasikan pada *SubBytes*, tahap selanjutnya yaitu menggeser atau melakukan transposisi baris *state[]* sesuai yang telah ditentukan pada tahapan *ShiftRows*. Baris kedua akan bergeser satu kolom ke kiri, baris ketiga akan bergeser dua kolom ke kiri, dan baris keempat akan bergeser tiga kolom ke kiri. Untuk lebih jelasnya, berikut adalah proses pada tahapan *ShiftRows*. Pada tahapan *MixColumns* ini, masing-masing kolom *state[]* hasil proses tahap *ShiftRows* akan dikalikan dengan matriks *multiplication* seperti pada persamaan di bawah, dan akan disimpan kembali ke dalam *state[]* (S_4).

Selanjutnya kembali dilakukan tahap *AddRoundKey* dengan melakukan operasi XOR antara *state[]* hasil dari tahap *MixColumns* dengan *round key* yang ke-1. Berikut adalah proses *AddRoundKey* untuk ronde ke-1. Setelah itu tahapan-tahapan *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* diulangi sebanyak Nr-1. Karena kunci 256-bit memiliki nilai Nr sebanyak 14, maka tahap yang disebutkan sebelumnya diulang sebanyak 13 kali. Lalu pada ronde terakhir (ke-14) akan melakukan tahapan seperti sebelumnya, namun melewati tahap *MixColumns*. Hal yang sama dilakukan terhadap blok selanjutnya hingga seluruh blok *cipher* terenkripsi.

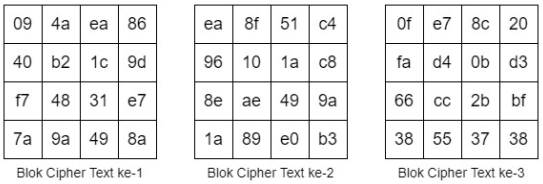
Berikut merupakan gambaran keseluruhan proses enkripsi *plain text* menggunakan algoritma AES yang ditunjukkan pada **Gambar 8**. Dan *cipher text* yang didapat adalah **"0940f77a4ab2489aea1c3149869de78aea968e1a8f10ae89511a49e0c4c89ab30ffa6638e7d4cc558c0b2b3720d3bf38"**.



Gambar 8. Proses enkripsi pada masing-masing blok

4.4. Deskripsi Menggunakan AES

Proses dekripsi tidak jauh berbeda dengan yang dilalui pada proses enkripsi, hanya prosesnya yang merupakan kebalikan dari proses enkripsi. Sama seperti pada proses enkripsi, *cipher text* perlu dibagi menjadi blok-blok 128-bit, seperti yang ditunjukkan pada gambar di bawah.



Gambar 9. Blok *cipher text*

Masing-masing blok akan melalui proses dekripsi (*invers*) yang terdapat pada AES, lalu hasil (*state[]*) proses tersebut akan di XOR dengan blok IV atau blok *cipher text* sebelumnya. Pada simulasi ini pertama kali akan melakukan proses dekripsi pada blok *cipher text* yang pertama.

Pada tahapan *AddRoundKey*, blok *cipher text* akan di XOR dengan *round key*. Berbeda dengan proses enkripsi, pada proses dekripsi ini dimulai dari *round key* yang ke Nr (ke-14) dan tahapannya mundur sampai ronde ke-0. Lalu tahap selanjutnya yaitu tahap *InwShiftRow*s Ronde ke-Nr. Proses pada tahapan ini merupakan kebalikan dari tahap *ShiftRows*, namun yang berbeda adalah pergeserannya kebalikan dari proses enkripsi, yaitu pada proses dekripsi akan bergeser ke kanan pada baris kedua, ketiga dan keempat.

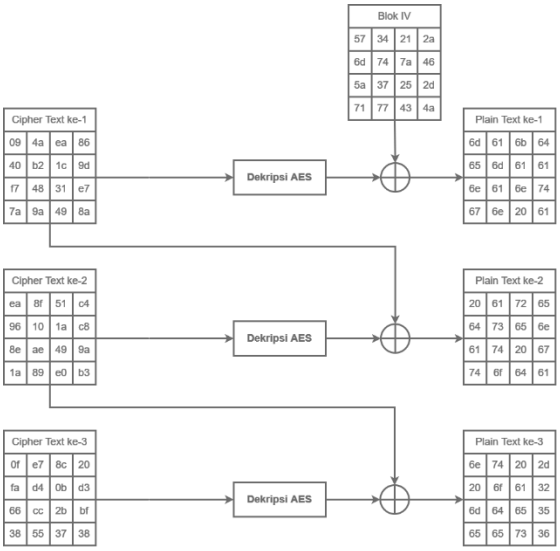
Tahap selanjutnya yaitu *InvSubBytes* Ronde ke-Nr. Pada tahapan ini dilakukan proses transformasi *state[]* hasil dari proses sebelumnya (*InvShiftRows*) atau S_2 sesuai dengan tabel *Invers* S-Box *byte* per *byte*. Berikut di bawah ini tabel *Invers* S-Box yang digunakan pada tahap dekripsi.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Gambar 10. Tabel *invers* S-Box

Tahap *AddRoundKey* saat ini sama dengan tahap *AddRoundKey* di awal, yaitu melakukan operasi XOR antara *state[]* dengan *round key* yang ke-13. Berikut adalah gambaran proses pada tahapan *AddRoundKey* untuk ronde ke-13. Kemudia tahap selanjutnya yaitu tahap *InvMixColumns*. Langkah pada tahapan ini hampir sama dengan tahapan *MixColumns*, namun yang membedakan pada tahap *InvMixColumns* adalah dengan mengalikan masing-masing kolom *state[]* dengan matriks *invers multiplication*.

Selanjutnya tahapan *AddRoundKey*, *InvMixColumns*, *InvShiftRows*, dan *InvSubBytes* akan diulangi sebanyak Nr-1 kali. Selanjutnya pada ronde terakhir (ke-0) hanya akan melalui tahap *AddRoundKey*. Dilakukan hal yang sama pula untuk blok-blok *cipher text* yang lain, lalu *state[]* dari hasil dekripsi menggunakan AES akan di XOR dengan IV (untuk blok *cipher* pertama) atau di XOR dengan blok *cipher* sebelumnya. Berikut adalah gambaran hasil dari proses dekripsi pada masing-masing blok *cipher* yang ditunjukkan pada **Gambar _** di bawah. Lalu untuk *plain text* yang didapat yaitu **"mengamankan data datastore dengan metode aes-256"**. Hasil dekripsi telah berhasil dan didapat nilai yang sama dengan *plain text* sebenarnya.



Gambar 11. Proses dekripsi pada masing-masing blok

4.5. Pengembangan Algoritma

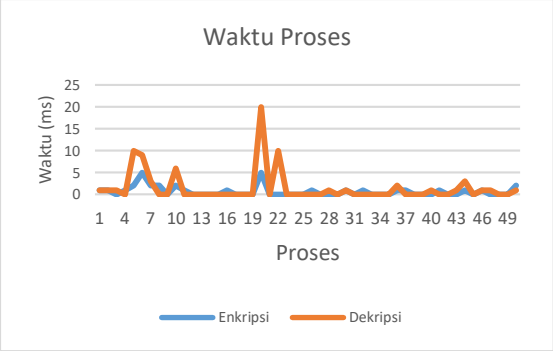
Pada tahapan ini akan dilakukan proses penulisan algoritma *Advanced Encryption Standard* (AES) ke dalam sebuah kode program menggunakan bahasa pemrograman Kotlin, sesuai dengan hasil analisis.

4.6. Pengujian Algoritma AES

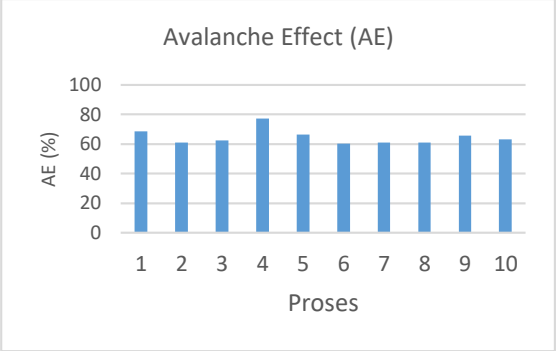
Proses yang dilakukan pada tahapan ini bertujuan untuk menguji efektifitas algoritma dengan cara mengukur kecepatan proses enkripsi beserta proses dekripsi. Selain itu dilakukan pula pengujian untuk mengukur keamanannya dengan mencari tingkat keacakan hasil *cipher text* menggunakan metode *Avalanche Effect.*

4.7. Pengukuran Kecepatan

Dalam pengukuran kecepatan digunakan fungsi bawaan dari bahasa pemrograman Kotlin, dengan perhitungan waktu proses berakhir dikurangi dengan waktu dimulainya proses enkripsi atau dekripsi. Dimana satuan yang digunakan yaitu mili-sekon (ms), yaitu sama dengan 0.001 detik. Adapun tahapan yang akan dilakukan yaitu dengan melakukan masing-masing proses yang diulangi sesuai dengan sampel *plain text* dan kunci, setelah itu di akhir akan diambil rata-rata dari proses enkripsi dan dekripsi. Adapun grafik hasil dari proses pengukuran ditunjukkan di bawah ini.



Gambar 12. Grafik hasil pengukuran kecepatan



Gambar 13. Grafik hasil dari tahap pengukuran *Avalanche Effect*

Lalu dari tahapan di atas, didapat hasil kecepatan prosesnya memiliki rata-rata **0.88 ms** pada proses enkripsi dan **0.44 ms** pada proses dekripsi. Waktu tertinggi untuk proses enkripsi berada di **15 ms**, sedangkan pada proses dekripsi berada di **6 ms**. Hasil ini lebih cepat dibandingkan dengan penelitian sebelumnya yang dilakukan oleh Elsa Elvira Awal dengan rata-rata 3,94 detik pada proses enkripsi dan rata-rata 3,99 detik pada proses dekripsi [10]. Selain itu hasil pengukuran kecepatan ini lebih cepat dibandingkan penelitian sebelumnya yang dilakukan oleh [23] yang memiliki nilai rata-rata sebesar 9,33 ms pada proses enkripsi dan 5,68 ms pada proses dekripsi.

4.8. *Avalanche Effect*

Kemudian terdapat tahapan *avalanche effect*. Pengujian ini bertujuan untuk menemukan tingkat keacakan dari nilai *cipher text* yang dihasilkan dari proses enkripsi. Skenario yang digunakan yaitu dengan cara menjalankan proses enkripsi pada sebuah *plain text*, selanjutnya *plain text* tersebut akan diubah satu karakter dan kembali dilakukan proses enkripsi. Dari kedua hasil tersebut akan dihitung persentase perubahan bit-nya. Sesuai dengan yang disampaikan sebelumnya, algoritma yang baik memiliki nilai persentase *Avalanche Effect* berada di angka minimal 50% (setengahnya) atau lebih besar [9], [17], [23], [24] berikut adalah persamaan untuk menghitung persentase *Avalanche Effect*.

$$AE = \frac{jumlah\ bit\ berubah}{jumlah\ bit\ total} \times 100\% \quad (1)$$

Untuk melakukan pengujian ini dibutuhkan bantuan perangkat lunak yang bernama *Cryptool* versi 2.1 untuk mengetahui jumlah perubahan bit antara dua *cipher text*.

Berikut adalah grafik hasil dari membandingkan dua *cipher text* yang telah dilakukan untuk mendapatkan nilai persentase *Avalanche Effect* dari algoritma AES yang telah dibuat.

Setelah melakukan perbandingan antara 10 pasang *cipher text*, didapatkan nilai rata-rata *Avalanche Effect* dari algoritma yang telah dikembangkan ini mencapai **64.68%**. Sesuai dengan yang disampaikan sebelumnya, algoritma yang baik memiliki nilai persentase minimal di angka 50% (setengahnya) atau lebih.

4.9. Implementasi Algoritma ke dalam Sistem

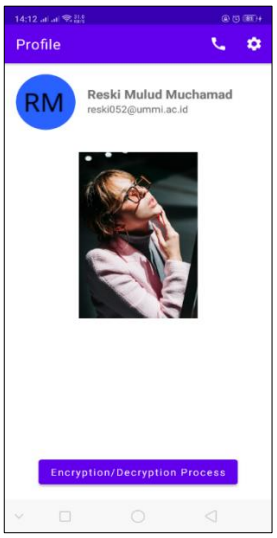
Setelah algoritma *Advanced Encryption Standard* (AES) dikembangkan serta dilakukan pengujian, hasil dari tahap tersebut akan diimplementasikan ke dalam sistem aplikasi Android, dan aplikasi tersebut dinamakan *Encrypted DataStore*. Aplikasi yang akan dikembangkan hanya berupa sampel yang bertujuan untuk menerima masukan dari pengguna, data tersebut selanjutnya akan disimpan ke dalam *DataStore*. Sebelum disimpan, data masukan akan dienkripsi menggunakan algoritma yang telah dikembangkan, lalu ketika data tersebut akan diakses akan melalui proses dekripsi terlebih dahulu.

Alat yang digunakan untuk mengembangkan aplikasi ini sepenuhnya menggunakan Android Studio. Aplikasi yang akan dikembangkan hanya memiliki dua halaman serta satu tampilan dialog yang menampilkan hasil proses enkripsi dan dekripsi. Halaman pertama yaitu halaman profil yang datanya diambil dari *DataStore*, dan halaman kedua adalah halaman pengaturan yang di dalamnya terdapat *form* untuk memasukkan data yang akan disimpan ke dalam *DataStore*. Berikut di bawah ini adalah penjelasan halaman hasil dari aplikasi yang telah dikembangkan dalam menerapkan algoritma AES ke dalam aplikasi Android.

1. Halaman Profil. Di dalam halaman ini menampilkan data atau informasi yang diambil dari *DataStore*. Data tersebut yaitu nama, email dan gambar yang diambil dari internet secara acak dengan menggunakan *API Key* (untuk mengakses gambar dari internet) yang diambil dari *DataStore*. Selain itu terdapat tombol telepon yang jika ditekan akan mengarahkan ke aplikasi pemanggil dan akan menampilkan nomor telepon sesuai dengan data yang

dimasukkan. Data yang ditampilkan merupakan data yang telah melalui proses dekripsi. Berikut adalah tampilan dari halaman profil yang telah dikembangkan.

2.



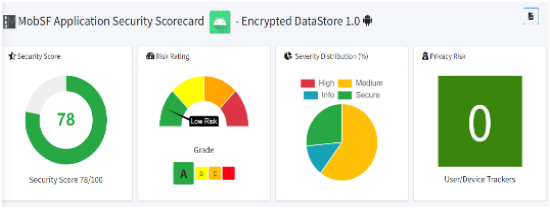
Gambar 14. Tampilan antar muka pada halaman profil

3. Halaman pengaturan (*setting*). Pada halaman ini menampilkan *form* masukan untuk menerima data dari pengguna. Data yang dimasukkan yaitu nama, email, nomor telepon dan *API Key* untuk mengakses gambar dari internet. Setelah menekan tombol "Submit", data yang dimasukkan akan dienkripsi sebelum disimpan ke dalam *DataStore*.
4. Dialog hasil proses enkripsi dan deskripsi. Pada dialog ini menampilkan hasil dari proses enkripsi dengan menunjukkan *cipher text*. Selain itu terdapat tombol untuk menampilkan hasil dari proses dekripsi dengan menunjukkan *plain text* hasil dekripsi dari *cipher text*.

4.10. Analisis Kemanan

Analisis keamanan dilakukan untuk mengetahui tingkat keamanan dan juga menguji kerentanan dari aplikasi Android yang telah dikembangkan. Untuk melakukan analisis keamanan, dibutuhkan bantuan perangkat lunak *Mobile Security Framework* (MobSF). Dan di bawah ini adalah skor hasil analisis keamanan aplikasi *Encrypted DataStore* yang dilakukan melalui perangkat lunak MobSF.

Berdasarkan hasil penilaian pada tahapan analisis keamanan menggunakan perangkat lunak MobSF, aplikasi *Encrypted DataStore* memiliki nilai keamanan yang baik dan memiliki tingkat risiko keamanan yang rendah, karena hanya ditemukan komponen yang memiliki level risiko di tingkat medium dan tidak ditemukan komponen yang memiliki level risiko di tingkat tinggi (*high*). Oleh karena itu aplikasi *Encrypted DataStore* mendapatkan skor 78/100 dan berada di *grade* A seperti yang ditunjukkan pada **Gambar 15**.



Gambar 15. Skor hasil analisis kemanan menggunakan MobSF

5. KESIMPULAN DAN SARAN

Berdasarkan hasil dari serangkaian tahap penelitian yang telah dilakukan dalam mengembangkan serta menerapkan algoritma *Advanced Encrypted Standard* pada aplikasi berbasis Android, maka dapat disimpulkan bahwa setelah dilakukan seluruh proses enkripsi dan dekripsi, didapatkan hasil kecepatan prosesnya kurang lebih 0.88 ms pada proses enkripsi dan 0.44 ms pada proses dekripsi. Setelah itu hasil pengujian *Avalanche Effect* mendapatkan nilai rata-rata sebesar 64.68% dan dapat dikatakan baik karena memiliki nilai persentase AE yang lebih dari 50%.

Penerapan algoritma *Advanced Encryption Standard* ke dalam sistem aplikasi Android telah memberikan keamanan terhadap data yang disimpan ke dalam DataStore dengan berhasil mengenkripsi datanya. Setelah menerapkan algoritma ke dalam sistem aplikasi Android dan dilakukan analisis keamanan, didapatkan hasil bahwa aplikasi yang telah dibuat memiliki risiko keamanan yang rendah dengan mendapatkan skor 78/100 (*grade* A).

Adapun saran yang diberikan untuk penelitian lebih lanjut di masa mendatang adalah sebagai berikut:

1. Mendukung enkripsi data selain data teks (string). Selain tipe datanya, diharapkan ke depannya dapat mendukung enkripsi data yang disimpan menggunakan *DataStore* tipe *Proto DataStore.*
2. Karena hasil penelitian ini akan dipublikasikan sebagai pustaka sumber terbuka (*open source*), diharapkan bagi pengembang Android dapat berkontribusi untuk meningkatkan kualitas hasil penelitian ini.

DAFTAR PUSTAKA

[1] W. K. Pertiwi, "Jumlah Pengguna Ponsel di Dunia Tembus 5 Miliar," *Kompas*, 2021. https://tekno.kompas.com/read/2021/09/02/09144137/jumlah-pengguna-ponsel-di-dunia-tembus-5-miliar (diakses Jan 13, 2022).

[2] D. Curry, "Android Statictics (2022)," *Businness of Apps*, 2022. https://www.businessofapps.com/data/android

-statistics/ (diakses Sep 07, 2022).
[3] R. Ario, "Jumlah Unduhan Aplikasi di Google Play Capai 28 Miliar," *Timlo*, 2020. https://timlo.net/baca/115867/jumlah-unduhan-aplikasi-di-google-play-capai-28-miliar (diakses Sep 07, 2022).
[4] Android Developer, "App Architecture: Data Layer - DataStore," 2022. https://developer.android.com/topic/libraries/architecture/datastore (diakses Sep 07, 2022).
[5] N. Matondang, I. N. Isnainiyah, dan A. Muliawatic, "Analisis Manajemen Risiko Keamanan Data Sistem Informasi (Studi Kasus: RSUD XYZ)," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 2, no. 1, hal. 282–287, Apr 2018, doi: 10.29207/resti.v2i1.96.
[6] Android Public Tracker, "Feature Request: Support for encryption in Datastore," *Issue Tracker Google*, 2021. https://issuetracker.google.com/issues/167697691 (diakses Sep 07, 2022).
[7] M. E. Smid, "Development of the Advanced Encryption Standard," *J. Res. Natl. Inst. Stand. Technol.*, vol. 126, no. 2, hal. 126024, Agu 2021, doi: 10.6028/jres.126.024.
[8] N. Anwar, M. Munawwar, M. Abduh, dan N. B. Santosa, "Komparatif Performance Model Keamanan Menggunakan Metode Algoritma AES 256 bit dan RSA," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 2, no. 3, hal. 783–791, Des 2018, doi: 10.29207/resti.v2i3.606.
[9] A. P. Putra, H. Herfina, S. Maryana, dan A. Setiawan, "Implementasi Algoritma AES (Advance Encryption Standard) Rijndael Pada Aplikasi Keamanan Data," *JIPETIK J. Ilm. Penelit. Teknol. Inf. Komput.*, vol. 1, no. 2, hal. 46–51, 2020.
[10] E. E. Awal *et al.*, "Analisis Perbandingan Hasil Enkripsi dan Deskripsi Algoritma Kriptografi Rijndael dan Twofish untuk Penyandian Data," *J. Mhs. Ilmu Komput.*, vol. 03, no. 01, hal. 1–6, 2022.
[11] D. A. Meko, "Perbandingan Algoritma DES, AES, IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data," *J. Teknol. Terpadu*, vol. 4, no. 1, hal. 8–15, Jul 2018, doi: 10.54914/jtt.v4i1.110.
[12] I. Gunawan, *Keamanan Data: Teori dan Implementasi*, 1 ed. Sukabumi: Jejak Publisher, 2021.
[13] I. Y. Sari *et al.*, *Keamanan Data dan Informasi*. Yayasan Kita Menulis, 2020.
[14] H. Mukhtar, *Kriptografi Untuk Keamanan Data*, 1 ed. Yogyakarta: Deepublish, 2018.
[15] N. Setiawan, "Kasus Kejahatan Siber Pada Telepon Seluler Android," *Cyber Secur. dan Forensik Digit.*, vol. 2, no. 1, hal. 24–29, 2019, doi: 10.14421/csecurity.2019.2.1.1420.
[16] M. O. Kadang, *Algoritma dan Pemrograman: Buku Bahan Ajar*, 1 ed. Makassar: Humanities Genius, 2021.
[17] R. R. Fauzi dan T. Wellem, "Perancangan Kriptografi Block Cipher berbasis Pola Dribbling Practice," *Aiti*, vol. 18, no. 2, hal. 158–172, 2021, doi: 10.24246/aiti.v18i2.158-172.
[18] A. Nugrahantoro, A. Fadlil, dan I. Riadi, "Optimasi Keamanan Informasi Menggunakan Algoritma Advanced Encryption Standard (AES) Mode Chiper Block Chaining (CBC)," *J. Ilm. FIFO*, vol. 12, no. 1, hal. 12, Jul 2020, doi: 10.22441/fifo.2020.v12i1.002.
[19] R. Damanik, P. D. . Silitonga, dan W. Ginting, *Membangun Aplikasi Android dengan Database SQLite*, 1 ed. Yayasan Kita Menulis, 2020.
[20] G. Gunawan, S. M. Damanik, F. B. Larasati, A. F. Zuri, dan S. Solikhun, *Dasar - Dasar Pemrograman Android*, 1 ed. Yayasan Kita Menulis, 2021.
[21] H. Herlinah dan M. KH, *Pemrograman Aplikasi Android dengan Android Studio, Photoshop, dan Audition*, 1 ed. PT Elek Media Komputindo, 2019.
[22] J. Daemen dan V. Rijmen, *The Design of Rijndael: The Advanced Encryption Standard (AES)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020.
[23] F. Rizky, "Implementasi Kriptografi Dengan Metode Advanced Encryption Standard (AES) Untuk Realtime Chat Berbasis Mobile Pada E - Learning Politeknik Negeri Lhokseumawe," *JAISE (Journal Artif. Intell. Softw. Eng.*, vol. 1, no. 2, hal. 1–8, 2019, doi: 10.30811/jaise.v1i2.2520.
[24] I. Fitriani dan A. B. Utomo, "Implementasi Algoritma Advanced Encryption Standard (AES) pada Layanan SMS Desa," *JISKA (Jurnal Inform. Sunan Kalijaga)*, vol. 5, no. 3, hal. 153–163, Nov 2020, doi: 10.14421/jiska.2020.53-03.