PROPOSAL PENELITIAN
KUALIFIKASI

MODIFIKASI PROSES ENKRIPSI DAN DEKRIPSI ALGORITMA AES BERBASISKAN *FINITE STATE MACHINE* PADA FPGA

Oleh:

Fauziah

99218021

PROGRAM DOKTOR TEKNOLOGI INFORMASI

UNIVERSITAS GUNADARMA

2022

DAFTAR ISI

1. PENDAHULUAN

1.1 Latar belakang

Perkembangan Teknologi Informasi yang begitu pesat hari ini sangat membantu kehidupan manusia. Teknologi informasi saat ini semakin canggih, tuntutan akan kemudahan mencari informasi menjadi faktor penggerak pesatnya pertumbuhan teknologi informasi. Perkembangan teknologi internet juga tumbuh cukup pesat, hal ini mendukung untuk memudahkan masyarakat dalam mengakses informasi yang diinginkan. Data merupakan salah satu hal yang sangatlah penting sehingga diperlukakan adanya keamanan data dalam proses pengiriman ataupun penerimaan data yang dilakukan melalui media komputer. Pengamanan data ini digunakan untuk menjaga kerahasiaan, keutuhan, keabsahan dan ketersediaan data. Keamanan data dapat terpenuhi dengan melakukan enkripsi dan dekripsi.

Proses enkripsi pesan atau informasi dilakukan agar tidak dapat dipahami atau dibaca tanpa bantuan pengetahuan atau alat khusus disebut dengan kunci. Proses mengembalikan informasi yang sudah dienkripsi tersebut menjadi suatu informasi yang dapat dipahami kembali disebut dengan dekripsi (Primartha R, 2011). Kriptografi ada dua jenis, antara lain kriptografi klasik dan modern. Pengguna lebih mengandalkan kelebihan kriptografi modern dalam proses pengamanan data, tetapi pengguna kriptografi klasik pun masih banyak dengan melakukan kombinasi antara dua algoritma kriptografi klasik (Sadikin, 2012).

Proses enkripsi atau dekripsi banyak sekali pengembangan terhadap algoritma yang dapat digunakan dalam kedua proses tersebut. Algoritma kriptografi tersebut dapat menggunakan *Advanced Encryption Standard* (AES). Algoritma AES adalah blok *chipertext* simetrik yang dapat mengenkripsi (*encipher*) dan dekripsi (*decipher*) informasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut *ciphertext*, dekripsi adalah merubah *ciphertext* data menjadi bentuk semula yang kita kenal sebagai *plaintext.*

LI Zhen-rong dan kawan-kawan (ZHUANG Yi-qi, ZHANG Chao dan JIN Gang, 2009) mengimplementasikan AES untuk berdaya rendah dan biaya murah

dengan mengusulkan rancangan Zigbee *system-on-a-chip* (SoC). Biaya dan konsumsi daya dari AES yang diusulkan menggunakan komponen yang minimal dengan mengoptimalkan arsitektur SubBytes/InvSubBytes dan MixColumns/InvMixColumns. Metode tersebut mengintegrasikan prosedur enkripsi dan dekripsi bersama-sama dengan metode berbagi sumber daya, dan menggunakan strategi manajemen daya hierarkis berdasarkan *Finite State Machine* (FSM) dan teknologi *Clock Gating* (CG). Berdasarkan SMIC 0,18 µm teknologi *complementary metal oxide semiconductor* (CMOS), skala chip AES hanya sekitar 10,5 kGate, konsumsi daya yang sesuai adalah 69,1 W/MHz, dan throughput adalah 32 Mb/s, yang masuk akal dan cukup untuk sistem Zigbee. Dibandingkan dengan desain lain, arsitektur tesebut dapat direkomendasikan dalam penggunaan daya yang lebih rendah dan penggunaan komponen yang minimal, serta memanfaatkan sistem Zigbee dan perangkat portabel lainnya.

Liling Dong dan kawan-kawan (Ning Wu dan Xiaoqiang Zhang, 2015) melakukan penelitian yaitu, merancangan dan implementasi tiga struktur jalur data dari sirkuit enkripsi AES yaitu 128, 32 dan 8 bit menjadi tiga *state machine* dengan metode *Finite State Machine* (FSM), *Look-Up Table* (LUT), dan *Decoder-Switch-Encoder* (DSE), serta dianalisis kinerja area, frekuensi, dan konsumsi daya seluruh energi. Hasil percobaan menunjukkan bahwa rangkaian enkripsi AES dengan struktur jalur data 8 bit mencapai area terkecil, sedangkan rangkaian dengan struktur jalur data 128 bit melakukan *throughput* tertinggi dan konsumsi daya rata-rata terendah. Tiga struktur jalur data yang berbeda, s*tate machine* LUT mencapai kinerja konsumsi daya yang rendah. *State Machine* dengan metode DSE selalu digunakan untuk menerapkan optimasi konsumsi daya rendah dari S-box, tidak dapat memenuhi kebutuhan daya rendah, sehingga kinerja daya rendah dari metode DSE harus dikombinasikan dengan nonlinier dari rangkaian logika kombinasional asli.

Peneliti Sadegh Attari, Aein Rezaei Shahmirzadi dan kawan – kawan (Sadegh Attari, Aein Rezaei Shahmirzadi dan Mahmoud Salmasizadeh, Iman Gholampour, 2017) melakukan penelitian dengan mengimplementasi FPGA pada algoritma AES yang memiliki dua lapisan resistansi terhadap analisis serangan daya dan menerapkan konsep FSM yang dilengkapi dengan pembangkit bilangan acak.

Serangan urutan pertama dapat dicegah dan jumlah daya yang dibutuhkan untuk serangan urutan kedua berhasil meningkat dan koefisien korelasi menurun, seperti harapan.

Penelitian yang dilakukan oleh Veronica Ernita Kristianti dan kawan – kawan (Veronica, dkk, 2019) mengembangkan algoritma DES dengan menerapkan metode 8 putaran dan 2 fungsi cipher sehingga waktu proses enkripsi lebih optimal. Proses enkripsi data dilakukan secara paralel dengan menggunakan metode tersebut. Desain optimasi algoritma DES diimplementasikan dalam bahasa pemrograman VHDL pada perangkat FPGA XC3S1200E. Hasil pengujian menunjukkan kecepatan proses enkripsi adalah 9 *clock* tanpa latensi. Penggunaan sumber daya yang dihasilkan dari penerapan metode dalam penelitian ini adalah 1% *slices of Flip-Flop* dan *Latch*, 2% *slices of LUTs*, 71% *slices of bonded IOBs* dan 12% of GCLKs.

Penelitian ini mengusulkan metode enkripsi dan dekripsi algoritma AES dengan penerapan konsep FSM yang dapat diimplementasikan pada perangkat keras menggunakan FPGA dengan sumber daya dan komponen yang minimal.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dapat dirumuskan batasan masalah sebagai berikut:

1. Bagaimana mengembangkan metode enkripsi dan dekripsi algoritma AES dengan konsep FSM?
2. Bagaimana proses implementasi *realtime* ke dalam FPGA?

1.3. Batasan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dapat dirumuskan batasan masalah sebagai berikut :

1. Pesan dan kunci yang digunakan sebesar 128 bit.

2. Implementasi menggunakan perangkat lunak vivado.

1.4. Tujuan Penelitian

Sesuai dengan masalah penelitian yang telah diuraikan sebelumnya, maka tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Menghasilkan pengembangan metode enkripsi dan dekripsi algoritma AES dengan konsep FSM.
2. Mengimplementasikan rancangan pengembangan algoritma AES pada FPGA.

1.5. Manfaat dan Kontribusi Penelitian

Dari segi keilmuan, penelitian ini memberikan kontribusi berupa pengembangan metode enkripsi dan dekripsi dengan algoritma AES yang mengandung informasi rahasia. Dari sisi metode, penelitian ini diusahakan untuk menemukan cara baru/penambahan/modifikasi dari metode yang ada untuk proses enkripsi dan dekripsi dengan memanfaatkan hardware FPGA. Dari sisi teknologi, penelitian ini menghasilkan suatu rancangan algoritma AES dengan konsep FSM yang dapat membantu dan memudahkan pengguna dalam mengenkripsi serta mendekripsi pesan yang mengandung informasi rahasia.

2. TINJAUAN PUSTAKA

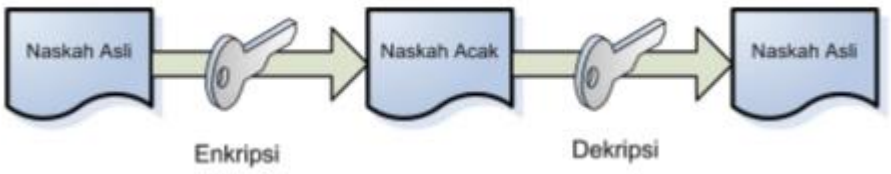
Bab ini menguraikan tentang studi literatur terkait dengan enkripsi pesan dan perkembangan penelitian mengenai proses enkripsi khususnya yang membahas penelitian - penelitian tentang enkripsi pesan rahasia dengan memakai perangkat keras FPGA yang telah dilakukan sejumlah peneliti.

2.1. Kriptografi

Kriptografi merupakan proses menyembunyikan informasi dengan menggunakan ilmu matematika untuk mengenkripsi dan dekripsi data. Hal ini memungkinkan untuk menyimpan informasi sensitif atau mengirimkannya melalui jaringan yang tidak aman (seperti Internet) sehingga tidak dapat dibaca oleh siapa pun kecuali penerima yang dituju (Shraddha Soni, dkk, 2012).

Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci).

Dalam kriptografi klasik, teknik yang biasa digunakan adalah enkripsi simetris dimana kunci dekripsi sama dengan kunci enkripsi. Untuk *public key cryptography*, diperlukan teknik enkripsi asimetris dimana kunci dekripsi tidak sama dengan kunci enkripsi. Enkripsi, dekripsi dan pembuatan kunci untuk teknik enkripsi asimetris memerlukan komputasi yang lebih intensif dibandingkan enkripsi simetris, karena enkripsi asimetris menggunakan bilangan - bilangan yang sangat besar. Namun, walaupun enkripsi asimetris lebih "mahal" dibandingkan enkripsi simetris, *public key cryptography* sangat berguna untuk *key management* dan *digital signature*.



Gambar 2.1 Proses enkripsi dan dekripsi (Sentot Kromodimoeljo, 2009)

Secara garis besar, proses enkripsi adalah proses pengacakan naskah asli (plaintext) menjadi naskah acak (ciphertext) yang sulit untuk dibaca oleh seseorang yang tidak mempunyai kunci dekripsi. Yang dimaksud dengan sulit untuk dibaca disini adalah probabilitas mendapat kembali naskah asli oleh seseorang yang tidak mempunyai kunci dekripsi dalam waktu yang tidak terlalu lama adalah sangat kecil. Jadi suatu proses enkripsi yang baik menghasilkan naskah acak yang memerlukan waktu yang lama untuk didekripsi oleh seseorang yang tidak mempunyai kunci dekripsi. Satu cara untuk mendapatkan kembali naskah asli tentunya dengan menerka kunci dekripsi, jadi proses menerka kunci dekripsi harus menjadi sesuatu yang sulit. Tentunya naskah acak harus dapat didekripsi oleh seseorang yang mempunyai kunci dekripsi untuk mendapatkan kembali naskah asli. Walaupun awalnya kriptografi digunakan untuk merahasiakan naskah teks, kini kriptografi digunakan untuk data apa saja yang berbentuk digital (Sentot Kromodimoeljo, 2009).

2.2. Kriptografi Block Cipher

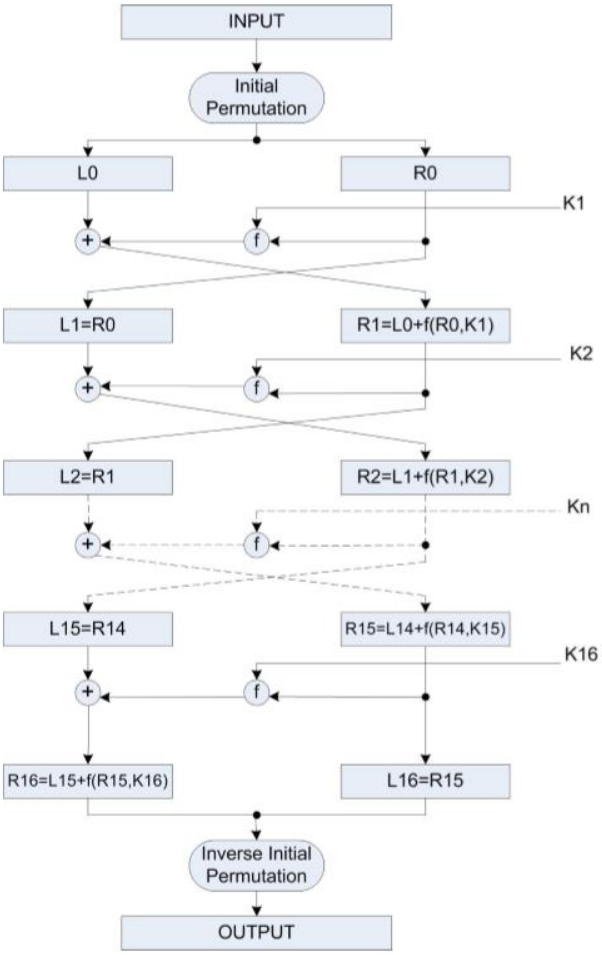
Menurut Claude Shannon, yang dianggap bapak dari teori informasi, dalam paper yang diterbitkannya tahun 1949, menganjurkan dua metode untuk mempersulit analisa statistik: diffusion dan confusion. Analisa statistik kerap berandalkan pengetahuan mengenai struktur statistik dari naskah asli dan sisa dari struktur ini dalam naskah acak. Untuk enkripsi sangat sederhana seperti Caesar cipher, sisa dari struktur ini masih sangat besar, bahkan masih utuh dan terlokalisir, jadi sangat mudah untuk mencari struktur dalam naskah acak. Efek diffusion bertujuan memperlemah sisa struktur dengan menyebarnya secara merata ke bagian yang cukup besar dari naskah acak meliputi banyak karakter. Struktur statistik (dalam teori informasi kerap disebut redundancy) masih ada tetapi sudah tersebar. Satu - satunya cara menghilangkan struktur tanpa menghilangkan informasi adalah dengan kompresi. Analisa statistik juga mencari hubungan antara kunci dengan naskah acak. Untuk simple substitution cipher, pengetahuan a priori mengenai statistik naskah asli memperkecil ruang kunci yang perlu diselidiki secara drastis, karena dapat digunakan untuk menghubungkan kunci dengan struktur statistik dari naskah acak. Efek confusion bertujuan untuk menghilangkan atau membuat tidak

jelas hubungan antara naskah acak dengan kunci. Block cipher seperti DES/3DES, CAST, IDEA dan AES menggunakan efek diffusion dan confusion untuk mempersulit analisa statistic.

2.2.1 Algoritma *Data Encryption Standard* (DES)

DES (Data Encryption Standard) pertama dijadikan standard FIPS (Federal Information Processing Standards) oleh NIST (National Institute of Standards and Technology) tahun 1977 untuk digunakan oleh semua instansi pemerintahan Amerika Serikat, dan semua kontraktor dan penyedia jasa untuk pemerintahan Amerika Serikat. DES dirancang oleh tim IBM yang dipimpin Horst Feistel dengan bantuan dari NSA (National Security Agency). DES adalah teknik enkripsi pertama (selain *one-time pad*) yang tahan terhadap *linear cryptanalysis* dan *differential cryptanalysis*.

DES menggunakan kunci sebesar 64 bit untuk mengenkripsi blok sebesar 64 bit. Akan tetapi karena 8 bit dari kunci digunakan sebagai parity, kunci efektif hanya 56 bit. Dalam DES, penomoran bit adalah dari kiri kekanan dengan bit 1 menjadi most significant bit, jadi untuk 64 bit, bit 1 mempunyai nilai 2^{63}. Permutasi menggunakan initial permutation dilakukan terhadap input sebesar 64 bit. Hasil permutasi dibagi menjadi dua blok L0 dan R0, masing-masing sebesar 32 bit, dimana L0 merupakan 32 bit pertama dari hasil permutasi dan R0 merupakan 32 bit sisanya (bit 33 hasil permutasi menjadi bit 1 R0). Sebanyak 16 putaran enkripsi dilakukan menggunakan fungsi cipher f dan setiap putaran menggunakan kunci 48 bit yang berbeda dan dibuat berdasarkan kunci DES. Efeknya adalah setiap blok secara bergantian dienkripsi, masing-masing sebanyak 8 kali (Sentot Kromodimoeljo, 2009).



Gambar 2.2 Enkripsi DES (Sentot Kromodimoeljo, 2009)

Gambar 2.2 secara garis besar menunjukkan proses enkripsi DES. Pada setiap putaran, blok sebesar 32 bit dienkripsi menggunakan rumus:

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n) \tag{2.1}$$

dan blok juga sebesar 32 bit tidak dienkripsi:

$$L_n = R_{n-1} \tag{2.2}$$

dimana :

L_n-1 adalah blok yang sedang giliran tidak dienkripsi,

$\oplus$ adalah operasi exclusive or secara bitwise,

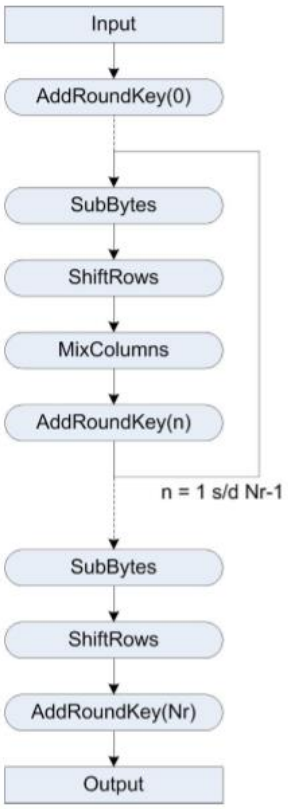
f adalah fungsi cipher yang akan dijelaskan,

R_n-1 adalah blok yang sedang giliran dienkripsi, dan

K_n adalah kunci untuk putaran n.

2.2.2 Algoritma *Advanced Encryption Standard* (AES)

AES (Advanced Encryption Standard) adalah teknik enkripsi yang dijadikan standard FIPS oleh NIST tahun 2001. AES dimaksudkan akan, secara bertahap menggantikan DES sebagai standard enkripsi di Amerika Serikat untuk abad ke 21. (DES sebagai standard FIPS telah dicabut, Mei 2005). AES menjadi standard melalui proses seleksi. Dari beberapa teknik enkripsi yang dicalonkan untuk menjadi AES, yang terpilih adalah enkripsi Rijndael. Teknik enkripsi ini termasuk jenis block cipher seperti halnya dengan DES. Perbedaan utama antara teknik enkripsi AES dan teknik enkripsi DES adalah AES juga menggunakan substitusi (menggunakan S-boxes) secara langsung terhadap naskah, sedangkan substitusi S-box digunakan DES hanya dalam fungsi cipher f yang hasilnya kemudian dioperasikan terhadap naskah menggunakan exclusive or, jadi DES tidak menggunakan substitusi secara langsung terhadap naskah. AES juga menggunakan kunci enkripsi yang lebih besar yaitu 128 bit, 192 bit, atau 256 bit (Sentot Kromodimoeljo, 2009).



Gambar 2.3 Enkripsi AES (Sentot Kromodimoeljo, 2009)

Gambar 2.3 secara garis besar, menunjukkan enkripsi AES. Input berupa naskah asli sebesar 128 bit, sedangkan output adalah naskah acak sebesar 128 bit. Setiap transformasi dilakukan secara langsung terhadap naskah, mulai dengan transformasi AddRoundKey(0). Jadi setiap transformasi harus mempunyai inverse agar naskah acak dapat didekripsi. Pada putaran 1 sampai dengan Nr−1, transformasi SubBytes, ShiftRows, MixColumns dan AddRoundKey dilakukan terhadap naskah. Pada putaran terahir (Nr), transformasi SubBytes, ShiftRows dan AddRoundKey dilakukan terhadap naskah (transformasi MixColumns tidak dilakukan).

Jumlah putaran (Nr) tergantung pada besar kunci yang digunakan. Tabel 2.1 menunjukkan jumlah putaran (Nr) untuk kunci sebesar 128 bit, 192 bit dan 256 bit. Jadi untuk kunci sebesar 128 bit, besar kunci (Nk) adalah 4 word (setiap word mempunyai 32 bit), besar blok (Nb) adalah 4 word, dan jumlah putaran (Nr) adalah 10.

Tabel 2.1. Tabel untuk Jumlah Putaran (Sentot Kromodimoeljo, 2009)

	Besar Kunci (Nk) dalam words	Besar Blok (Nb) dalam words	Jumlah Putaran (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

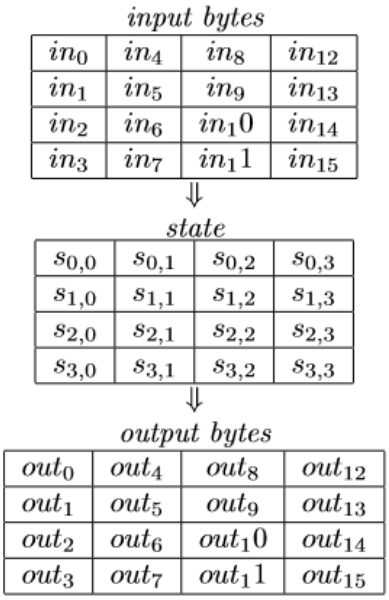
Penjelasan AES menggunakan konvensi urutan indeks untuk bit dan byte sebagai berikut:

- Urutan indeks bit sebagai input adalah $input_0$, $input_1$, $input_2$, dan seterusnya.
- Urutan indeks byte sebagai input adalah a_0, a_1, a_2, dan seterusnya.
- Dalam byte, bit menggunakan urutan indeks berlawanan dengan input: b_7, b_6, b_5, dan seterusnya sampai dengan b_0. Most significant bit dalam byte adalah b_7.

Tabel 2.2 Index bit dan byte AES (Sentot Kromodimoeljo, 2009)

input	0	1	2	3	4	5	6	7	8	9	...					...
a	0								1							...
b	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Tabel 2.2 memperlihatkan konvensi indeks bit dan byte untuk AES. Tabel menunjukkan bahwa untuk byte kedua (a_1), $b_7 = input_8$, $b_6 = input_9$, $b_5 = input_{10}$, dan seterusnya sampai dengan $b_0 = input_{15}$. Algoritma enkripsi AES beroperasi terhadap state dari naskah yang dipandang sebagai matrik terdiri dari 16 byte. Setiap kolom dari state merepresentasikan satu word, dengan $s_{0,i}$ sebagai most significant byte untuk kolom i. Gambar 2.4 menunjukkan bagaimana state didapat dari input dan bagaimana state dijadikan output. Transformasi AddRoundKey, SubBytes, ShiftRows dan MixColumns semua dilakukan terhadap state.



Gambar 2.4 Input, State dan Output (Sentot Kromodimoeljo, 2009)

Beberapa transformasi yang dilakukan dalam enkripsi AES memperlakukan byte seolah polynomial dalam polynomial field GF(2^8). Setiap bit dalam byte merepresentasikan koefisien suku polynomial sebagai berikut:

- bit b_7 merupakan koefisien untuk X^7,
- bit b_6 merupakan koefisien untuk X^6,
- dan seterusnya sampai dengan bit b_0 yang merupakan koefisien untuk X^0 (konstan).

Operasi inverse polynomial terhadap byte dapat dilakukan (dalam GF(2^8)) jika byte ≠ 0. Irreducible polynomial yang digunakan AES untuk GF(2^8) adalah

$$X^8 + X^4 + X^3 + X + 1$$

AES juga melakukan transformasi affine (dalam GF(2)) terhadap byte menggunakan rumus

$$b'_i = b_i + b_{i+4 \bmod 8} + b_{i+5 \bmod 8} + b_{i+6 \bmod 8} + b_{i+7 \bmod 8} + c_i \quad (2.3)$$

dengan $0 \leq i \leq 7$ dan byte c = 01100011. Jika dijabarkan, rumus menjadi

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Operasi SubByte (substitusi S-box) terhadap byte terdiri dari operasi inverse polynomial (jika byte ≠ 0) diikuti oleh transformasi affine menggunakan rumus 2.3. Jika byte = 0 maka hanya transformasi affine yang dilakukan, menghasilkan 01100011 (63 dalam notasi hexadecimal). Operasi SubByte (substitusi S-box) dan inverse operasi SubByte dalam bentuk tabel dibawah ini :

Tabel 2.3 S-BOX AES (Sentot Kromodimoeljo, 2009)

	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

Operasi AddRoundKey melakukan bitwise exclusive or menggunakan kunci putaran sebesar 128 bit terhadap state (yang juga 128 bit). Untuk setiap putaran, kunci putaran berbeda dari key schedule yang didapat dari ekspansi kunci enkripsi digunakan. Total ada Nr + 1 kunci putaran yang digunakan, satu untuk operasi awal

sebelum putaran pertama, dan satu untuk setiap putaran. Jadi ekspansi kunci menghasilkan Nb(Nr+1) = 4(Nr+1) words menggunakan algoritma sebagai berikut:

1. I ← 0,
2. w[i] ← [k[4i], k[4i+1], k[4i+2], k[4i+3]],
3. i ← i + 1, jika i < Nk kembali ke langkah 2,
4. t ← w[i−1],
5. jika i mod Nk = 0 maka t ← SubWord(RotWord(t)) ⊕ Rcon[i/Nk], kalau tidak, jika Nk > 6 dan i mod Nk = 4 maka t ← SubWord(t),
6. w[i] ← w[i−Nk] ⊕ t,
7. i ← i + 1, jika i < Nb(Nr +1) kembali ke langkah 4

dimana

1. k adalah kunci enkripsi,
2. w adalah array terdiri dari Nb(Nr +1) words dengan indeks mulai dari 0,
3. ⊕ adalah operasi bitwise exclusive or,
4. SubWord adalah operasi terhadap word dimana substitusi SubByte dilakukan terhadap setiap byte dalam word,
5. RotWord adalah operasi terhadap word dimana urutan byte dalam word diubah sebagai berikut:

   $$[a_0, a_1, a_2, a_3] \Rightarrow [a_1, a_2, a_3, a_0],$$

   dan
6. Rcon[j] adalah word sebagai berikut:

   $$[X^{j-1}, 0, 0, 0]$$

   dimana X^0 = 00000001, X = 00000010, X^2 = 00000100 dan seterusnya sampai dengan X^7 = 10000000, dan untuk X dengan pangkat > 7 aritmatika polynomial field digunakan.

Jadi Nk words pertama untuk w didapatkan langsung dari kunci enkripsi, sedangkan words selanjutnya untuk w ditentukan oleh word 1 posisi sebelumnya dan word Nk posisi sebelumnya. Kunci putaran diambil dari w sebagai berikut:

$$ki \leftarrow [w[4i], w[4i+1], w[4i+2], w[4i+3]]$$

dengan $0 \leq i \leq Nr$. Transformasi AddRoundKey(i) terhadap state adalah sebagai berikut:

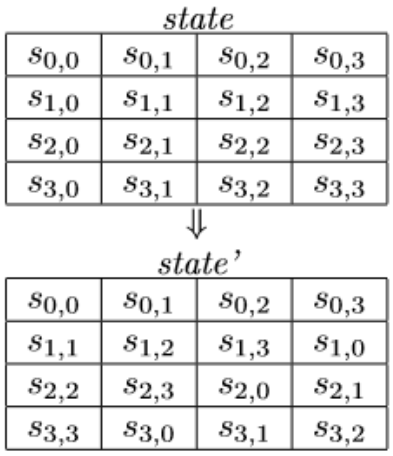
$$state' \leftarrow state \oplus ki.$$

Transformasi AddRoundKey mempunyai inverse yaitu transformasi AddRoundKey sendiri (ingat sifat operasi exclusive or).

Untuk transformasi SubBytes, substitusi SubByte dilakukan terhadap setiap byte dalam state. Untuk inverse transformasi SubBytes, substitusi dilakukan menggunakan inverse SubByte.

Transformasi ShiftRows menggeser byte dalam baris state sebagai berikut :

7. baris 0 tidak digeser,
8. baris 1 digeser 1 posisi kekiri dengan byte terkiri menjadi byte terkanan,
9. baris 2 digeser 2 posisi kekiri, jadi dua byte sebelah kiri ditukar dengan dua byte sebelah kanan, dan
10. baris 3 digeser 3 posisi kekiri, yang efeknya sama dengan menggeser 1 posisi kekanan.



Gambar 2.5 ShiftRows ((Sentot Kromodimoeljo, 2009)

Gambar 2.5 memperlihatkan efek ShiftRows terhadap state. Tidak terlalu sulit untuk menunjukkan bahwa transformasi ShiftRows mempunyai inverse. Transformasi terahir yang perlu dijelaskan untuk enkripsi AES adalah transformasi MixColumns. Untuk MixColumns, kolom dalam state (yang merupakan word) diperlakukan sebagai polynomial dengan koefisien dalam GF(2^8). Word dibawah ini :

$$[a_0, a_1, a_2, a_3]$$

diperlakukan sebagai polynomial

$$a_3X^3 + a_2X^2 + a_1X + a_0.$$

Pertambahan polynomial dilakukan sebagaimana pertambahan dalam polynomial ring. Agar hasil tetap sebesar word, perkalian dilakukan modulo polynomial

$$g(x) = X^4 + 1.$$

Tidak terlalu sulit untuk menunjukkan bahwa

$$X^i \bmod (X^4 + 1) = X^{i \bmod 4} \tag{2.4}$$

karena −1 = 1 dalam GF(2^8). Karena X4 + 1 bukan merupakan irreducible polynomial dalam K[X] dimana K = GF(2^8), maka K[X]/g(X)K[X] bukan merupakan field: tidak semua polynomial mempunyai inverse. Akan tetapi polynomial

$$a(x) = 03X^3 + 01X^2 + 01X + 02 \tag{2.5}$$

mempunyai inverse

$$a^{-1}(x) = 0\mathrm{b}X^3 + 0\mathrm{d}X^2 + 09X + 0\mathrm{e}. \tag{2.6}$$

Transformasi MixColumns mengalikan (modulo polynomial g(X)) setiap kolom dalam state (diperlakukan sebagai polynomial) dengan polynomial a(X). Transformasi MixColumns terhadap state dapat dirumuskan efeknya terhadap setiap kolom c sebagai berikut:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

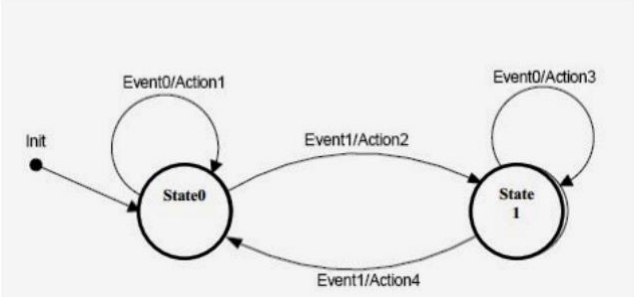
Transformasi MixColumns mempunyai inverse yaitu mengalikan setiap kolom dalam state dengan $a^{-1}(X)$ modulo polynomial g(X). Karena semua transformasi yangdilakukandalam enkripsi AES mempunyai inverse, dekripsi dapat dilakukan menggunakan inverse transformasi dengan urutan kebalikan dari urutan transformasi enkripsi. Proses dekripsi dimulai dengan transformasi AddRoundKey(Nr) terhadap naskah acak, diikuti dengan transformasi inverse ShiftRows, inverse SubBytes dan seterusnya.

AES telah menggantikan DES sebagai standard enkripsi untuk keperluan instansi pemerintahan Amerika Serikat. Ada yang meragukan ketangguhan AES karena semua transformasi yang dilakukan dalam enkripsi AES mempunyai rumus aljabar yang elegan. Akan tetapi rumus yang elegan tidak berarti parameter kunci dapat dipecahkan dengan mudah dari persamaan. Kriptografi public key menggunakan rumus aljabar yang elegan, namun parameter kunci privat sulit untuk dipecahkan. Kembali ke AES, karena enkripsi adalah manipulasi berbagai bit, pemecahan parameter kunci dapat dirumuskan sebagai masalah SAT dalam aljabar

Boolean, namun masalah SAT adalah masalah yang bersifat NP-complete yang pada umumnya tidak dapat dipecahkan secara efisien (untuk pemecahan kunci enkripsi AES, ruang pencarian terlalu besar sehingga dalam prakteknya pemecahan tidak dapat dilakukan).

2.3. FSM (Finite State Machine)

Finite State Machine (FSM) merupakan sebuah mesin abstrak yang berfungsi untuk mendefinisikan sekumpulan kondisi yang menentukan kapan suatu state harus berubah. Setiap state yang sedang dijalankan tersebut menentukan perilaku yang terjadi pada objek yang bersangkutan. Diagram state FSM digambarkan pada gambar 1 sebagai berikut, (Feisal, 2015).



Gambar 2.5 Skema Finite State Machine

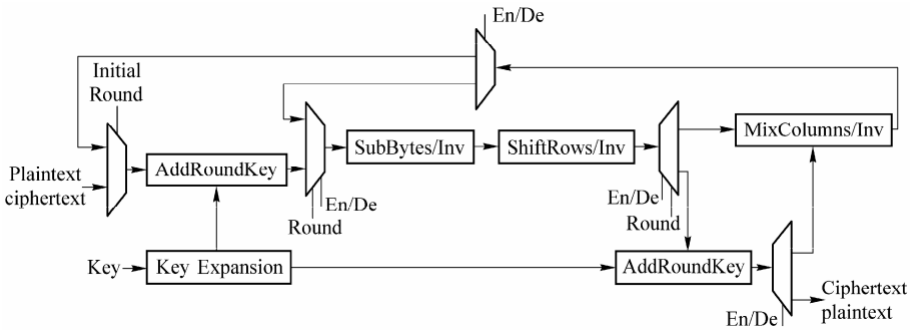
Diagram tersebut memperlihatkan FSM dengan dua buah state dan dua buah input serta empat buah aksi output yang berbeda : seperti terlihat pada gambar, ketika sistem mulai dihidupkan, sistem akan bertransisi menuju state0, pada keadaan ini sistem akan menghasilkan Action1 jika terjadi masukan Event0, sedangkan jika terjadi Event1 maka Action2 akan dieksekusi kemudian sistem selanjutnya bertransisi ke keadaan State1 dan seterusnya (Setiawan, 2016).

2.4 Perkembangan Penelitian Terkait

2.4.1 Implementasi AES pada Sistem Chip Zigbee

Penelitian (LI Zhen-rong, dkk, 2009) mengimplementasikan AES agar berdaya rendah dan biaya murah dengan mengusulkan rancangan Zigbee system-on-a-chip (SoC). Biaya dan konsumsi daya dari AES yang diusulkan menggunakan

komponen yang minimal dengan mengoptimalkan arsitektur SubBytes / InvSubBytes dan MixColumns/InvMixColumns terlihat pada gambar 2.6.



Gambar 2.6 Arsitektur Perangkat Keras Enkripsi/Dekripsi

Metode tersebut mengintegrasikan prosedur enkripsi dan dekripsi bersama-sama dengan metode berbagi sumber daya, dan menggunakan strategi manajemen daya hierarkis berdasarkan Finite State Machine (FSM) dan teknologi Clock Gating (CG). Berdasarkan SMIC 0,18 μm teknologi complementary metal oxide semiconductor (CMOS), skala chip AES hanya sekitar 10,5 kGate, konsumsi daya yang sesuai adalah 69,1 W/MHz, dan throughput adalah 32 Mb/s, yang masuk akal dan cukup untuk sistem Zigbee. Dibandingkan dengan desain lain, arsitektur tesebut dapat direkomendasikan dalam penggunaan daya yang lebih rendah dan penggunaan komponen yang minimal, serta memanfaatkan sistem Zigbee.

2.4.2 Implementasi State Machine pada Enkripsi AES

Penelitian yang telah dilakukan (Liling Dong, dkk, 2015) merupakan penelitian yang merancangan dan implementasi tiga struktur jalur data dari sirkuit enkripsi AES. Jalur data AES yang digunakan yaitu 128, 32 dan 8 bit menjadi tiga *state machine* dengan mengimplementasi *Finite State Machine* (FSM), *Look-Up Table* (LUT), dan *Decoder-Switch-Encoder* (DSE), serta dianalisis kinerja area, frekuensi, dan konsumsi daya seluruh energi.

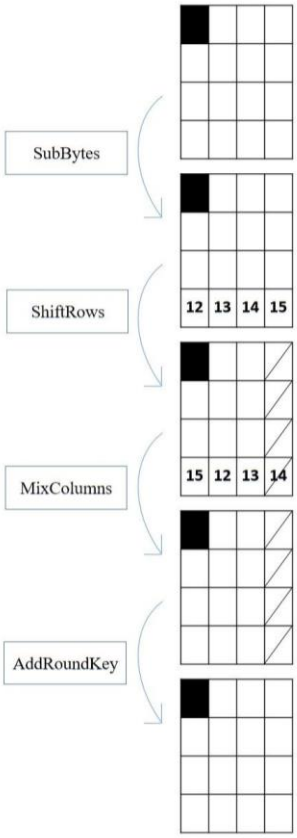
Jalur Data (bit)	Implementasi	Area (μm^2)	f_{max} (MHz)	Daya (μW)
128	FSM	309,36	3,88	3,88
	LUT	262,79	38,45	3,72
	DSE	572,14	38,3	5,07
32	FSM	435,76	38,51	5,35
	LUT	482,33	38,2	3,91
	DSE	1520,16	36,7	5,48
8	FSM	798,34	37,74	10,26
	LUT	908,11	37,48	7,42
	DSE	417,88	35,97	10,34

Hasil percobaan menunjukkan bahwa rangkaian enkripsi AES dengan struktur jalur data 8 bit mencapai area terkecil, sedangkan rangkaian dengan struktur jalur data 128 bit melakukan *throughput* tertinggi dan konsumsi daya rata-rata terendah. Tiga struktur jalur data yang berbeda, s*tate machine* LUT mencapai kinerja konsumsi daya yang rendah. *State Machine* dengan metode DSE selalu digunakan untuk menerapkan optimasi konsumsi daya rendah dari S-box, tidak dapat memenuhi kebutuhan daya rendah, sehingga kinerja daya rendah dari metode DSE harus dikombinasikan dengan nonlinier dari rangkaian logika kombinasional asli.

2.4.3 Implementasi Algoritma AES Pada FPGA

Penelitian (Sadegh Attari, dkk, 2017) mengimplementasi FPGA pada algoritma AES yang memiliki dua lapisan resistansi terhadap analisis serangan daya dan menerapkan konsep FSM yang dilengkapi dengan pembangkit bilangan acak.

AES adalah algoritma kunci simetris yang mendukung ukuran kunci 128, 192, dan 256 bit. Panjang input (teks biasa) adalah 128-bit yang terstruktur sebagai matriks 4 × 4 byte di mana setiap byte mewakili nilai dalam GF(2^8). Untuk menghasilkan keluaran (ciphertext), AES bekerja di beberapa putaran serupa. Satu putaran dari AES ditunjukkan pada Gambar 2.7.



Gambar 2.7 Satu Putaran pada AES

Setiap putaran memiliki empat utama operasi. Operasi tersebut teridir dari langkah *AddRoundKey* dengan menggunakan fungsi xor antara plaintext dan kunci. *SubByte* adalah substitusi non-linear, yang diterapkan pada setiap *byte* dari *state* secara mandiri. Tabel *S-box* dapat dibalik dan dibuat oleh menggabungkan dua konversi, perkalian *invers* dan *affine transformasi*. *ShiftRow*s hanya menggeser baris ke kiri berdasarkan *indeks* mulai dari nol. *MixColumns* dianggap sebagai polinomial atas GF (2^8) dan dikalikan dengan polinomial tetap (‘03’x^3 + ‘01’x^2 + ‘01’x + ‘02’.) modul x^4+1. Dua langkah terakhir digunakan untuk difusi yang lebih baik dan SubByte membuat lebih banyak kebingungan untuk algoritma.

Pada putaran pertama, fungsi AddRoundKey tambahan diterapkan dan pada putaran terakhir MixColumns tidak diterapkan. Nomor putaran tergantung pada panjang kunci dan berbeda dari satu kunci ke lain. Skema kunci 128bit memiliki 10 putaran. Kunci bulat adalah output dari proses yang disebut jadwal kunci. Masukan dari proses ini adalah kunci 128-bit awal. Jadwal utama terdiri dari dari dua bagian;

pertama panjang kunci bulat ditentukan kemudian kunci 128-bit awal harus diperluas ke ukurannya. Panjang balok 128 dan 10 putaran, 1408 bit *Roundkey* yang diperlukan.

Penelitian menghasilkan pada serangan urutan pertama dapat dicegah dan jumlah daya yang dibutuhkan untuk serangan urutan kedua berhasil meningkat dan koefisien korelasi menurun.

2.4.4 Implementasi Algoritma DES Pada FPGA

Penelitian yang dilakukan oleh Veronica Ernita Kristianti dan kawan – kawan (Veronica, dkk, 2019) mengembangkan algoritma DES dengan menerapkan metode 8 putaran dan 2 fungsi cipher sehingga waktu proses enkripsi lebih optimal. Proses enkripsi data dilakukan secara paralel dengan menggunakan metode tersebut. Desain optimasi algoritma DES diimplementasikan dalam bahasa pemrograman VHDL pada perangkat Field-Programmable Gate Array (FPGA) XC3S1200E. Hasil pengujian menunjukkan kecepatan proses enkripsi adalah 9 clock tanpa latensi. Penggunaan sumber daya yang dihasilkan dari penerapan metode dalam penelitian ini adalah 1% slices of Flip-Flop and Latch, 2% slices of LUTs, 71% slices of bonded IOBs and 12% of GCLKs.

2.5 Perbandingan Penelitian

Penelitian-penelitian terkait pencarian dokumen gambar yang mengandung informasi teks disajikan pada tabel 2.8.

Tabel 2.8. Ringkasan Penelitian Enkripsi dan Dekripsi Pesan FPGA

No	Peneliti / Judul	Metode	Hasil	Keterbatasan
1	LI Zhen-rong, ZHUANG Yi-qi, ZHANG Chao, dan JIN Gang, 2009. “Low-power and area-optimized VLSI implementation of AES coprocessor for Zigbee system”	Algoritma AES dengan FSM dan Clock Gating .	Berdasarkan SMIC 0,18 μm teknologi complementary metal oxide semiconductor (CMOS), skala chip AES hanya sekitar 10,5 kGate,	

			konsumsi daya yang sesuai adalah 69,1 W/MHz, dan throughput adalah 32 Mb/s, yang masuk akal dan cukup untuk sistem Zigbee. Arsitektur tesebut dapat direkomendasikan dalam penggunaan daya yang lebih rendah dan penggunaan komponen yang minimal, serta memanfaatkan sistem Zigbee.	
2	Liling Dong, Ning Wu, and Xiaoqiang Zhan, 2015. "Low Power State Machine Design for AES Encryption Coprocesso"	Algoritma AES dengan metode FSM, LUT dan DSE.	Arsitektur enkripsi AES dengan struktur jalur data 8 bit mencapai area terkecil, sedangkan rangkaian dengan struktur jalur data 128 bit melakukan throughput tertinggi dan konsumsi daya rata-rata terendah. Tiga struktur jalur data yang berbeda, state machine LUT mencapai kinerja konsumsi daya yang rendah. State	metode yang diusulkan hanya fokus terhadap area dan throughput.

			Machine dengan metode DSE selalu digunakan untuk menerapkan optimasi konsumsi daya rendah dari S-box, tidak dapat memenuhi kebutuhan daya rendah, kinerja daya rendah dari DSE harus dikombinasikan dengan non-linier dari rangkaian logika kombinasional asli.	
3	Sadegh Attari, Aein Rezaei Shahmirzadi, Mahmoud Salmasizadeh, Iman Gholampour, 2017. "Finite State Machine Based Countermeasure for Cryptographic Algorithms".	Algoritma enkripsi AES dan konsep FSM.	Teknik yang diusulkan menghasilkan serangan urutan pertama dapat dicegah dan jumlah daya yang dibutuhkan untuk serangan urutan kedua berhasil meningkat dan koefisien korelasi menurun.	
4	Veronica Ernita, Eri Prasetyo Wibowo, Atit Pertiwi, Busono Soerowirdjo and Hamzah Afandi, 2019.	Algoritma DES	Hasil pengujian menunjukkan kecepatan proses enkripsi adalah 9 clock tanpa latensi.	

	“Implementation Optimization Of The Des Algorithm On Fpga To Support Smartcard Processors”.		Penggunaan sumber daya yang dihasilkan dari penerapan metode dalam penelitian ini adalah 1% slices of Flip-Flop and Latch, 2% slices of LUTs, 71% slices of bonded IOBs and 12% of GCLKs.	

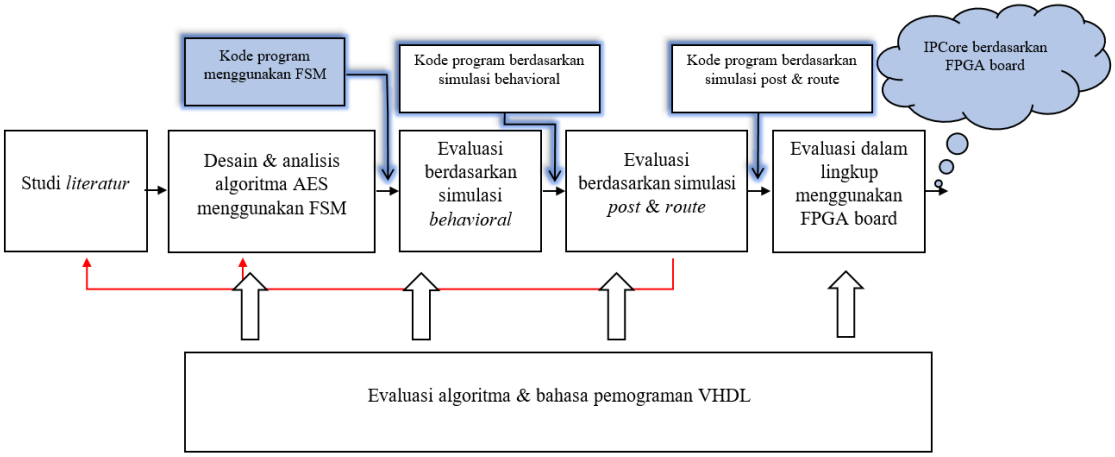
3. Metodologi Penelitian

3.1 Tahapan Penelitian

Penelitian ini merupakan pengembangan dari penelitian terdahulu yaitu dengan memodifikasi algoritma enkripsi-dekripsi dengan konsep *finite state machine* (FSM). Tahapan penelitian ini terbagi menjadi empat tahapan dan ditunjukan oleh gambar 3.1. Tahap awal dilakukan dengan studi *literature* berupa pencarian studi pustaka seperti artikel-artikel dan buku mengenai materi-materi yang lebih mendalam dari berbagai konsep algoritma enkripsi-dekripsi kedalam bentuk konsep *finite state machine* (FSM) serta berkonsultasi dengan pakar. Tujuan dari studi *literature* adalah untuk menganalisis kelebihan dan kekurangan algoritma tersebut, sehingga dapat mengusulkan konsep dan algoritma enkripsi-dekripsi baru dalam bentuk konsep FSM serta mengembangkan metode yang dapat diimplementasikan dalam rangkaian FPGA.

Tahap kedua adalah mendesain dan menganalisis algoritma enkripsi-dekripsi AES menggunakan konsep *finite state machine* (FSM). Pada tahapan ini dilakukan dengan menggunakan perangkat lunak Xilinx ISE dan melakukan modifikasi setiap proses algoritma enkripsi-dekripsi AES dalam bentuk konsep FSM, sehingga dapat diimplementasi kedalam perangkat lunak. Tahap ketiga melakukan evaluasi berdasarkan simulasi behavioral, pengujian dilakukan dengan menggunakan pemograman VHDL dan beberapa data *numerical*.

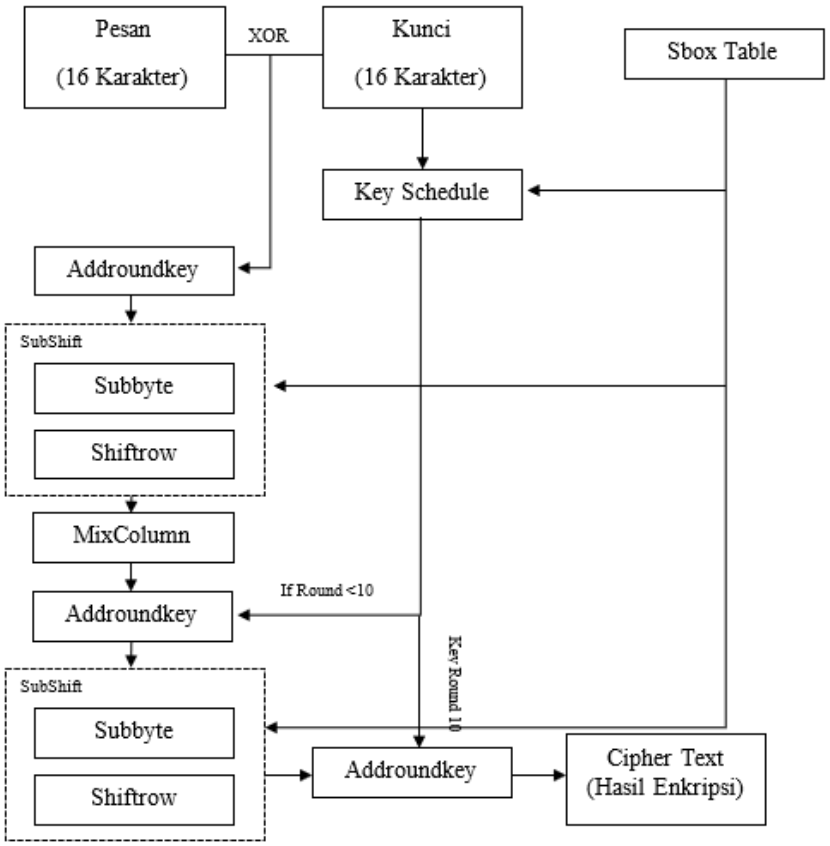
Tahap keempat yaitu evaluasi berdasarkan simulasi post-route merupakan pengujian yang dilakukan untuk mengetahui kode program sudah benar sehingga dapat menampilkan sinyal masukkan dan keluaran hasil proses simulasi. Tahap terakhir merupakan evaluasi dalam lingkup menggunakan FPGA board, pengijian dilakukan dengan menggunakan FPGA board yang keluarannya berupa dekripsi komponen yang dipakai pada algoritma AES yang termodifikasi dalam bentuk konsep FSM sehingga menghasilkan IPCore berdasarkan FPGA board.



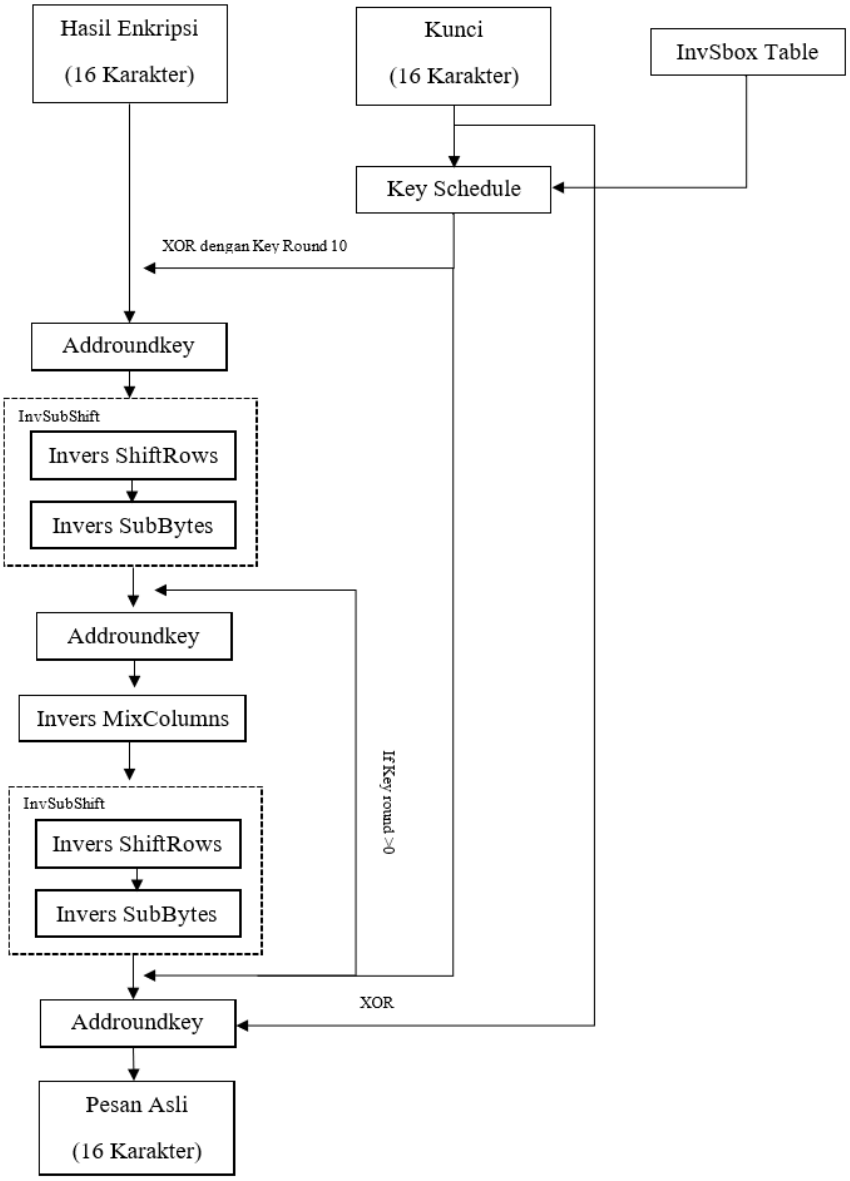
Gambar 3.1 Metodelogi penelitian

3.2 Disain dan Analisis

Penelitian yang telah dilakukan terdahulu, metode yang digunakan untuk memberikan keamanan tingkat tinggi terhadap peretasan dengan melakukan paralelisasi antara proses transformasi subbyte dan shiftrows. Penelitian ini mengajukan untuk memodifikasi proses urutan algoritma AES dengan konsep FSM, sehingga dapat membuat langkah kerja pada setiap proses enkripsi-deskripsi dalam bentuk *state*. Pada pengembangan yang dilakukan dalam penelitian ini, berharap agar dapat mengurangi penggunaan komponen sehingga dapat lebih efisien dan mengoptimalkan kecepatan yang tinggi. Diagram alur proses enkripsi dan dekripsi dapat terlihat pada gambar 3.2 dan gambar 3.3.



Gambar 3.2 Diagram alur Proses Enkripsi



Gambar 3.3 Diagram alur Proses Deskripsi

Pada gambar 3.2 dan gambar 3.3 merupakan diagram alur penelitian algoritma AES yang telah dikembangkan pada penelitian sebelumnya. Berikut merupakan proses untuk algoritma AES adalah :

1. Proses perhitungan *key schedule*

   Proses pertama dalam algoritma AES adalah mencari *key schedule* yang akan digunakan untuk 10 putaran.

2. Proses XOR antara *plaintext* dengan *cipher key*

Proses kedua adalah melakukan XOR antara *plaintext* dengan *cipher key* untuk menghasilkan *addroundkey*.

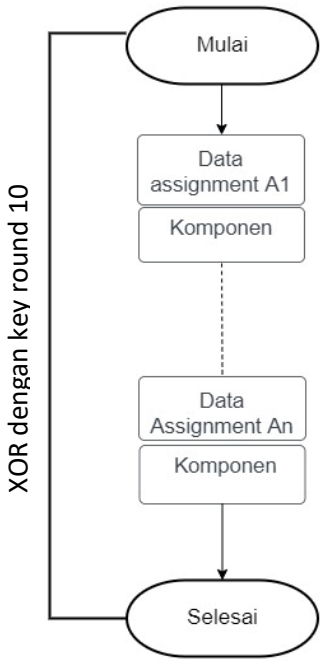
3. Proses Transformasi Subbytes, Shiftrows, Mixcolumn, dan Addroundkey

   Transformasi subbytes dilakukan dengan cara subtitusi dengan tabel S-Box yang sudah disimpan dalam memory. Data yang dihasilkan merupakan matriks 4x4. Pada transformasi shiftrows dilakukan pergeseran untuk baris 0 tidak terjadi pergeseran, untuk baris 1 matriks terjadi pergeseran 1 byte ke kiri, untuk baris 2 matriks terjadi pergeseran 2 byte ke kiri, dan untuk baris 3 matriks terjadi pergeseran 3 byte ke kiri. Pada transformasi mixcolumn mengalikan (modulo polynomial g(X)) setiap kolom dalam state (diperlakukan sebagai polynomial) dengan polynomial a(X). Pada transformasi addroundkey terjadi proses XOR antara state (matriks) dengan key schedule. Proses ini berulang sebanyak 9 putaran.

4. Proses Transformasi Subbytes, Shiftrows, dan addroundkey

   Pada proses ini hanya terdapat 3 proses transformasi yaitu subbytes, shiftrows dan addroundkey. Proses ini terjadi pada putaran terakhir dalam algoritma AES. Setelah proses ini selesai maka akan menghasilkan sebuah *cipher text*.

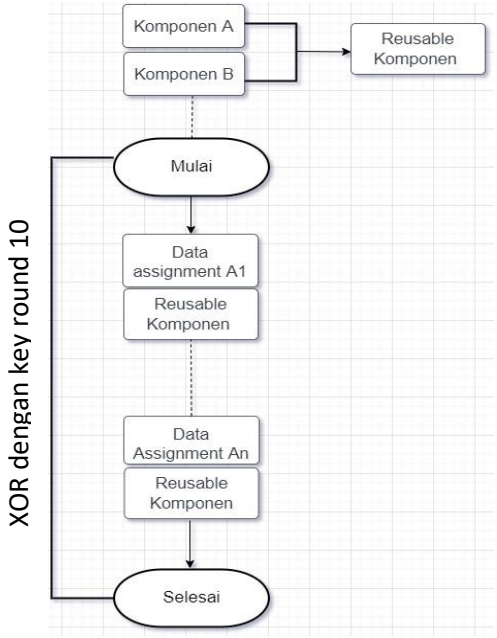
Desain pembuatan pada *key schedule* terdapat proses XOR antara plaintext dan *cipher key* sebanyak 10 *round key* sehingga menghasilkan *addroundkey*. Proses tersebut dapat dilihat pada gambar 3.4 sebagai berikut.



Gambar 3.4 Alur Proses Perhitungan Key Schedule

Gambar 3.4 merupakan alur pada proses perhitungan key schedule dengan menggunakan gerbang xor antara data *assignment* A1…An dengan *chiper key*. Xor merupakan gerbang logika yang dijadikan sebagai komponen pendukung dalam menentukan chiper key berikutnya. Penulis merancang desain untuk meminimalisasi penggunaan komponen xor dengan mengeluarkan komponen xor dan membuat menjadi dua komponen utama yaitu komponen A dan komponen B.

Komponen A terdiri dari proses xor antara data assignment A1…An, *chiper key* dan nilai Rcon. Komponen B terdiri dari proses xor antara data *assignment* A1…An dengan *chiper key*. Alur desain tersebut dapat dilihat pada gambar 3.5.



Gambar 3.5 Desain Komponen A dan Komponen B

Gambar 3.5 merupakan desain komponen A dan komponen B yang akan dijadikan komponen yang reusable. Komponen tersebut terdapat gerbang logika xor yang dapat dipakai untuk menentukan *key Schedule*. Dari hasil penelitian tersebut diusulkan untuk menambahkan konsep FSM pada setiap transpormasi. Transpormasi yang dibuat ada 3 buah yaitu : paralelisasi subbyte dan shiftrows, MixColumns, dan AddRoundKey

3.3 RENCANA KERJA

Rencana penyelesaian waktu dalam penelitian ini selama 12 bulan, dengan rincian kerja yang terlihat pada tabel 3.1.

Tabel 3.1. Rencana Pelaksanaan Penelitian

BULAN KE / LANGKAH PENELITIAN	1	2	3	4	5	6	7	8	9	10	11	12
Studi Pustaka												
Mendesain dan menganalisis Algoritma AES menggunakan FSM												
Mengevaluasi berdasarkan simulasi behavioral												
Mengevaluasi berdasarkan simulasi post dan route												
Mengevaluasi dalam lingkup menggunakan FPGA board												

DAFTAR PUSTAKA

LI Zhen-rong, ZHUANG Yi-qi, ZHANG Chao, JIN Gang, 2009. “Low-power and area-optimized VLSI implementation of AES coprocessor for Zigbee system”. The Journal of China Universities of Posts and Telecommunications.

Kromodimoeljo, Sentot. 2009. Teori dan Aplikasi Kriptografi. Penerbit: SPK IT Consulting.

Ferry Wahyu Wibowo, 2011. “Finite State Machine Untuk Pengendali Elevator Berbasis Field Programmable Gate Array”. Jurnal Dasi.

Liling Dong, Ning Wu, and Xiaoqiang Zhang, 2015. “Low Power State Machine Design for AES Encryption Coprocesso”. International MultiConference of Engineers and Computer Scientists.

Sadegh Attari, Aein Rezaei Shahmirzadi, Mahmoud Salmasizadeh, Iman Gholampour, 2017. “Finite State Machine Based Countermeasure for Cryptographic Algorithms”. IEEE.

Veronica Ernita Kristianti, Eri Prasetyo Wibowo, Atit Pertiwi, Busono Soerowirdjo and Hamzah Afandi, 2019. “Implementation Optimization Of The Des Algorithm On Fpga To Support Smartcard Processors”. ICIC Express Letters Part B: Applications.

E. A. Lee and S. A. Seshia, 2017. “Introduction to Embedded Systems - A Cyber-Physical Systems Approach Second Edition”. MIT Press.
Diakses dari :
https://ptolemy.berkeley.edu/books/leeseshia//

Umer Farooq, dan M. Faisal Aslam, 2016. “Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA”. Journal of King Saud University – Computer and Information Sciences.

Adnan Mohsin Abdulazeez, dan Ari Shawkat Tahir, 2013. “Design and Implementation of Advanced Encryption Standard Security Algorithm using FPGA”. International Journal of Scientific & Engineering Research.

Ahmed M. Atteya, dan Ahmed H. Madian, 2014. “A Hybrid Chaos-AES Encryption Algorithm and Its Impelmention Based on FPGA”. IEEE.

Feisal, ferdian, 2015. “Pengembangan FSM Untuk Memodelkan Agen Dan Pergerakan Olahraga Futsal.” Bogor: Institute Pertanian Bogor.

Setiawan, I. 2006. “Perancangan Software Embedded System Berbasis FSM.” Semarang: Universitas Diponegoro.