USULAN TUGAS AKHIR

IMPLEMENTASI ALGORITME AES 128BIT PADA MIKROKONTROLER NODEMCU MENGGUNAKAN ARSITEKTUR WEB SERVICE REST UNTUK KEAMANAN PENGIRIMAN DATA

Oleh :

RHOMY IDRIS SARDI

F1D 016 076

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MATARAM

2020

USULAN TUGAS AKHIR

IMPLEMENTASI ALGORITME AES 128BIT PADA MIKROKONTROLER NODEMCU MENGGUNAKAN ARSITEKTUR WEB SERVICE REST UNTUK KEAMANAN PENGIRIMAN DATA

Telah diperiksa dan disetujui oleh Tim Pembimbing :

1. Pembimbing Utama

Ariyan Zubaidi, S.Kom., M.T. Tanggal: 30 April 2020
NIP. 19860913 201504 1 001

2. Pembimbing Pendamping

Andy Hidayat Jatmika, S.T., M.Kom. Tanggal: 21 April 2020
NIP. 19831209 201212 1 001

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Teknik
Universitas Mataram

Prof. Dr.Eng. I Gede Pasek Suta Wijaya, S.T., M.T.
NIP: 197311 200003 1 001

USULAN TUGAS AKHIR

IMPLEMENTASI ALGORITME AES 128BIT PADA MIKROKONTROLER NODEMCU MENGGUNAKAN ARSITEKTUR WEB SERVICE REST UNTUK KEAMANAN PENGIRIMAN DATA

Telah diperiksa dan disetujui oleh Tim Penguji :

1. Penguji 1

Ahmad Zafrullah Mardiansyah, S.T., M.Eng Tanggal: 06 Juni 2020
NIP. -

2. Penguji 2

Dr. Eng. I Gde Putu Wirarama W. W., S.T., M.T. Tanggal: 09 Juni 2020
NIP. 19840919 201803 1 001

3. Penguji 3

Ramaditia Dwiyansaputra, S.T., M.Eng Tanggal: 16 Juni 2020
NIP. -

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Teknik
Universitas Mataram

Prof. Dr.Eng. I Gede Pasek Suta Wijaya, S.T., M.T.
NIP: 197311 200003 1 001

DAFTAR ISI

COVER
LEMBAR PENGESAHAN
DAFTAR ISI iv
DAFTAR GAMBAR vi
DAFTAR TABEL vii
ABSTRAK viii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 4
1.5 Manfaat Penelitian 4
1.6 Sistematika Penulisan 4
BAB II TINJUAN DAN LANDASAN TEORI 6
2.1 Tinjauan Pustaka 6
2.2 Dasar Teori 7
2.2.1 Kriptografi 7
2.2.2 *Advanced Encryption Standard* (AES) - Rijndael 8
2.2.3 *Internet of Things* 11
2.2.4 Mikrokontroler NodeMCU 1.0 12
2.2.5 *Web Service* 13
2.2.6 REST 13
2.2.7 Flask 14
2.2.8 DHT-22 15
2.2.9 Arduino IDE 15
2.2.10 SQLite 17
2.2.11 Wireshark 17

BAB III METODOLOGI PENELITIAN 19
3.1 Diagram Alir Penelitian 19
3.2 Analisis Kebutuhan Sistem 21
3.2.1 Kebutuhan Perangkat Keras 21
3.2.2 Kebutuhan Perangkat Lunak 21
3.3 Perancangan Sistem 22
3.3.1 Perancangan Perangkat Keras 22
3.3.2 Perancangan Perangkat Lunak 23
3.3.3 Perancangan *Database* 25
3.3.4 Perancangan Algoritme 25
3.4 Implementasi Sistem 33
3.5 Pengujian dan Evaluasi Sistem 34
DAFTAR PUSTAKA 37

DAFTAR GAMBAR

Gambar 2. 1 Diagram proses enrkipsi AES 10
Gambar 2. 2 Ilustrasi Transformasi *ByteSub*() AES 11
Gambar 2. 3 Ilustrasi transformasi *ShiftRow*() AES 11
Gambar 2. 4 Ilustrasi transformasi *MixColumn*() AES 11
Gambar 2. 5 Ilustrasi transformasi *AddRoundKey*() AES 11
Gambar 2. 6 Mikrokontroler NodeMCU 1.0 13
Gambar 2. 7 Sensor DHT-22 15
Gambar 2. 8 Arduino IDE 16
Gambar 2. 9 Struktur Wireshark 18
Gambar 3. 1 Diagram Alir Penelitian 19
Gambar 3. 2 Rancangan proses komunikasi *client-server* REST 22
Gambar 3. 3 Rancangan alur kerja *client* REST 23
Gambar 3. 4 Rancangan alur kerja *server* REST 24
Gambar 3. 5 Rancangan *Database* 25
Gambar 3. 6 Pengindeks-an *bit* dalam *byte* pada blok 26
Gambar 3. 7 *State array* pada *input* dan *output* 26
Gambar 3. 8 *State array* ekivalen pada *word array* 26
Gambar 3. 9 Ilustrasi proses enkripsi AES 27
Gambar 3. 10 *AddRoundKey*[5] 28
Gambar 3. 11 *SubBytes*[5] 28
Gambar 3. 12 *ShitRows*[5] 29
Gambar 3. 13 Persamaan matriks *MixColumns*[5] 29
Gambar 3. 14 *MixColumns*[5] 29
Gambar 3. 15 Persamaan *Rcon* 30
Gambar 3. 16 Ilustrasi proses dekripsi AES 31
Gambar 3. 17 Persamaan matriks *MixColumns* 33
Gambar 3. 18 Skema Pengamanan data pada IoT 35

DAFTAR TABEL

Tabel 2. 1 Jumlah putaran setiap blok pada AES 8
Tabel 2. 2 Tabel S-box yang digunakan dalam transofrmasi *ByteSub*() AES 10
Tabel 2. 3 Spesifikasi perangkat NodeMCU 12
Tabel 3. 1 Kebutuhan perangkat keras sistem 21
Tabel 3. 2 Kebutuhan perangkat lunak sistem 21
Tabel 3. 3 Tabel S-Box^{-1} untuk transformasi *invers SubBytes* 32
Tabel 3. 4 Jadwal kegiatan 36

ABSTRAK

Di tengah tren *internet of things* (IoT) yang berkembang pesat seperti sekarang ini, terdapat beberapa tantangan yang harus dihadapi seperti masalah keamanan dan sumberdaya. Salah satu gangguan keamanan pada IoT adalah *interception* atau penyadapan data. Selain itu, IoT merupakan perangkat dengan kapasitas memori yang kecil dan termasuk perangkat dengan proses yang lambat. Menjawab tantangan tersebut, penelitian ini bertujuan melakukan pengujian terkait metode keamanan yang tepat dalam memenuhi unsur kerahasiaan data yang dikirimkan melaui jaringan nirkabel *wifi* oleh perangkat mikrokontroler NodeMCU menuju *server* basis data. Prosesor 80MHz dan memori dinamis 81920 *byte* yang dimiliki NodeMCU terbilang kecil sehingga dibutuhkan metode keamanan yang tepat digunakan pada perangkat minim *resource*. Salah satu metode keamanan yang *popular* adalah algoritme kriptografi AES. AES adalah algoritme yang ringan dan dapat digunakan pada *embedded system*. Data berbentuk plainteks akan dienkripsi menjadi *cipher text* menggunakan algoritme AES 128bit sebelum menuju *server* basis data. Pada *server*, data akan didekripsi menjadi bentuk semula kemudian disimpan ke dalam basis data. Proses pengiriman data menggunakan arsitektur *web service* REST (*Representational State Transfer*). Terdapat empat bagian pengujian untuk memastikan sistem telah berjalan dengan baik yakni pengujian fungsional, pengujian kinerja waktu dan mermori, serta pengujian keamanan.

Kata kunci : kriptografi, algoritme AES 128bit, *confidentiality*, NodeMCU, *web service* REST

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aspek keamanan merupakan salah satu unsur penting yang sering kali terlupakan dalam hal pengiriman data pada perangkat IoT. Tanpa adanya metode pengaman penyerang dapat mengambil data dengan melakukan *sniffing*. Pada penelitian[7] diperoleh bahwa data yang dikirim melalui jaringan nirkabel tanpa metode pengamanan berhasil di-s*niffing* menggunakan *tool* Wireshark. Penelitian[8] menyebutkan kebocoran informasi baik melalui akses yang tidak sah atau oleh orang dalam menyebabkan potensi ancaman. Misalnya, pada *smart farms* yang memanfaatkan perangkat cerdas untuk memantau dan menganalisis data yang dikumpulkan dapat digunakan di berbagai bidang penelitian seperti tanaman biologi dan genetika, ekonomi pertanian, perkiraan pasokan, dan prediksi penyakit. Perangkat yang terhubung ke jaringan memungkinkan penyerang dapat memanipulasi atau memperoleh informasi seperti keadaan tanah, tanaman, pembelian dan lain-lain yang dapat menyebabkan kerugian yang parah bagi petani, jika informasi tersebut digunakan oleh aktor atau aktor yang bermusuhan. Pada skala yang lebih besar, informasi agregasi *agriculture* penting di negara tertentu juga merupakan ancaman potensial. Dengan demikian keamanan data dan privasi adalah sebuah persyaratan yang sangat penting dan salah satu tujuan utama untuk memastikan operasi yang andal dalam ekosistem *agriculture* yang cerdas.

Berdasarkan standar *International Electrotechnical Commision* (IEC) 62591, enkripsi data pada sistem IoT merupakan salah satu unsur penting yang harus dipenuhi[18]. Pada buku panduan[6] sendiri mengatakan bahwa setiap orang yang tidak memiliki wewenang seharusnya tidak dapat melihat dan membaca data digital dalam sebuah sistem jaringan nirkabel. Selain aspek keamanan, penelitian[9] menyebutkan bahwa perangkat IoT adalah perangkat dengan kapasitas memori yang kecil sehingga dibutuhkan sebuah pengaman yang dirancang untuk penggunaan memori yang kecil, karena IoT termasuk perangkat dengan proses yang lambat.

Kerahasiaan data tak bisa terlepas dari kriptografi. Penelitian mengenai penerapan algoritme kriptografi untuk keamanan pengiriman data pernah dilakukan pada[7] yang menerapkan algoritme Lizard untuk enkripsi data pada mikrokontroler NodeMCU menggunakan arsitektur *web service* REST. Hasil pengujian menunjukkan bahwa algoritme Lizard berhasil memenuhi unsur keamanan data dengan proses enkripsi 216 dan 352 bit *plain text* membutuhkan waktu 0.02 dan 0.04 detik serta memori sebesar 0,3% dari memori total. Berdasarkan penelitian tersebut, diperoleh gagasan untuk melakukan penelitian dengan menerapkan algoritme kriptografi yang berbeda pada mikrokontroler berbasis Arduino.

Algoritme kriptografi yang cukup *popular* digunakan dalam keamanan data adalah AES (*Advanced Encryption Standard*). Selain itu algoritme AES adalah algoritme yang kuat, hal ini dibuktikan dari ditetapkannya algoritme AES oleh *National Institute of Standards and Technology* (NIST) pada November 2001 sebagai standar algoritme pengamanan di Amerika Serikat yang menggantikan algoritme *Data Encryption Standard* (DES), algoritme AES sendiri didesain sedemikian rupa sehingga cukup kuat terhadap kriptanalisis linier dan diferensial[14]. Berdasarkan referensi[17] juga bahwa algoritme AES memenuhi aspek keamanan, kemangkusan (*efficiency*), fleksibilitas, dan kebutuhan memori (penting untuk *embedded system*). Berlandaskan hal tersebut, algoritme AES dipilih untuk diterapkan pada penelitian ini.

Pada penelitian[21] menyebutkan data yang perlu dianalisis secara berkala pada *smart farms* dapat berupa data lingkungan seperti suhu, kelembaban dari tanah dan air, curah hujan, serta intensitas cahaya. Sebagai demonstrasi data yang digunanakan pada penelitian ini adalah data suhu dan kelembaban. Karena data tersebut termasuk data yang perlu dianalisis dan penting untuk *agriculture* seperti dijelaskan pada *poin* sebelumnya. Mikrokontroler berbasis arduino yang familiar digunakan saat ini dan dilengkapi perangkat untuk kebutuhan komunikasi melalui nirkabel adalah mikrokontroler NodeMCU dan Wemos. Pada penelitian ini dipilih mikrokontroler NodeMCU, karena berdasarkan referensi[22] diperoleh bahwa dari spesifikasi kedua

kontroler tersebut hampir sama dilihat dari prosesor dan *memory* yang digunakan berturut-turut 80MHz dan *flash* 4MB.

Penelitian ini berfokus pada implementasi algoritme AES 128 bit untuk keamanan data di perangkat mikrokontroler NodeMCU, khususya dalam enkripsi data yang akan dikirim dari mikrokontroler menuju *server* basis data. Setelah dilakukan proses implementasi algoritme tersebut, akan dilakukan pengujian fungsional, pengujian kinerja waktu dan mermori, serta pengujian keamanan yang nantinya akan ditarik kesimpulan dari hasil pengujian tersebut.

Sehingga berlandaskan pemaparan di atas, maka peneliti mengajukan proposal penelitian dengan menerapkan algoritme kriptografi AES 128bit pada mikrokontroler NodeMCU menggunakan arsitektur *web service* REST untuk keamanan pengiriman data.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka didapatkan rumusan masalah "Bagaimana kinerja algoritme kriptografi AES 128bit pada mikrokontroler NodeMCU menggunakan arsitektur *web service* REST untuk keamanan pengiriman data?".

1.3 Batasan Masalah

Pada penelitian ini, peneliti membuat batasan masalah untuk mencegah meluasnya ruang lingkup permasalahan dalam penelitian. Adapun batasan masalah tersebut diantaranya, yaitu:

1. Penelitian ini menggunakan algoritme kriptografi AES dengan panjang kunci 128bit untuk enkripsi dan dekripsi data.
2. Data yang diamankan pada penelitian ini berupa data suhu dan kelembaban
3. Pengujian keamanan dilakukan menggunakan metode *sniffing* pada pengiriman data (sistem *non-https*) yang dikirim dari mikrokontroler menuju *server* basis data menggunakan arsitektur WEB *Service* REST.
4. Algoritme kriptografi AES diimpelementasikan pada sisi *client* (*encryption*) dan sisi *server* (*decryption*).

5. Penelitian ini berfokus pada hasil pengujian fungsional, pengujian kinerja waktu dan memori yang dibutuhkan untuk proses enkripsi dan dekripsi serta keamanan.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah mengetahui kinerja algoritme kriptografi AES 128bit pada mikrokontroler NodeMCU menggunakan arsitektur *web service* REST untuk keamanan pengiriman data.

1.5 Manfaat Penelitian

Manfaat yang ingin didapatkan dari penelitian tugas akhir ini adalah:

1. Bagi Mahasiswa : dapat menerapkan ilmu yang didapat dari perkuliahan serta menambah wawasan mengenai penerapan algoritme kriptografi untuk keamanan pengiriman data pada IoT.
2. Bagi pembaca : menambah pengetahuan tentang keamanan pengiriman data pada IoT dan dapat digunakan sebagai referensi untuk penelitian selanjutnya.

1.6 Sistematika Penulisan

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut:

1. Bab I Pendahuluan

   Bagian bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.
2. Bab II Tinjauan dan Landasan Teori

   Bagian bab ini dibahas beberapa tinjauan pustaka berupa hasil penelitian sebelumnya yang digunakan sebagai acuan dalam penyusunan tugas akhir serta dipaparkan teori-teori pendukung yang berkaitan dengan penelitian.
3. Bab III Metodologi Penelitian

   Bagian bab ini berisi langkah penelitian, analisa kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis sistem.
4. Bab IV Hasil dan Pembahasan

Bagian bab ini membahas realisasi atau implementasi sistem kemudian membahas hasil pengujian sistem.

5. Bab V Kesimpulan

   Bagian bab ini merupakan bab terakhir yang memuat kesimpulan isi dari keseluruhan uraian yang ada pada bab-bab sebelumnya dan saran-saran dari hasil yang diperoleh yang nantinya diharapkan dapat bermanfaat dalam pengembangan selanjutnya.

BAB II
TINJUAN DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Setelah dilakukan telaah terhadap beberapa penelitian terdahulu, ada beberapa yang memiliki keterkaitan dengan penelitian yang dilakukan. Pada penelitian[7] menerapkan algroritma kriptografi Lizard menggunakan arsitektur *web service* REST pada perangkat NodeMCU. Penelitian ini bertujuan untuk memenuhi unsur kerahasiaan data yang dikirimkan oleh perangkat mikrokontroler NodeMCU dalam suatu lingkungan IoT. Berdasarkan hasil pengujian didapatkan bahwa algoritme Lizard berhasil memenuhi unsur kerahasiaan data dengan proses enkripsi 216 dan 352 bit *plain text* membutuhkan waktu 0.02 dan 0.04 detik serta memori sebesar 0,3% dari memori total.

Pada penelitian[10] menerapkan algoritme kriptografi AES dengan kunci 128bit pada komunikasi perangkat IoT tanpa menggunakan arsitektur *web service* REST. Penelitian ini bertujuan mengimplementasikan algoritme AES dengan kunci 128bit untuk memenuhi unsur kerahasiaan data serta mengetahui *resource* yang digunakan setelah penerapan algoritme AES. Berdasarkan hasil pengujian didapatkan bahwa algoritme AES dengan kunci 128bit dapat digunakan pada perangkat yang memiliki *resource* kecil serta memiliki kecepatan yang baik.

Pada penelitian[9] membandingkan algoritme kriptografi RSA dan AES untuk keamanan informasi perangkat ZigBee. Penelitian init bertujuan membandingkan kinerja kriptografi RSA (*Rivest Shamir Adleman*) dan AES (*Advanced Encryption Standard*) pada perangkat komunikasi ZigBee. Berdasarkan hasil pengujian didapatkan bahwa algoritme yang lebih baik digunakan pada perangkat komunikasi ZigBee adalah algoritme AES dengan waktu komputasi lebih cepat dan penggunaan memori lebih sedikit dibanding algoritme RSA.

Pada penelitian[3] menerapkan algoritme AES dengan kunci 128bit untuk enkripsi dan dekripsi *file* dokumen. Penelitian ini bertujuan mengimplementasikan algoritme AES dengan kunci 128 untuk mengamankan data dalam bentuk *file*

dokumen serta mengetahui kecepatan waktu yang dibutuhkan dalam proses enkripsi dan dekripsi. Berdasarkan hasil pengujian didapatkan bahwa algoritme AES dengan kunci 128bit berhasil memenuhi unsur keamanan data dan waktu yang dibutuhkan relatif cepat dibanding dengan penelitian sebelumnya[15]. Proses enkripsi untuk 7,1 MB dibutuhkan waktu 3,3 detik, dibandingkan dengan 1,8 MB dibutuhkan waktu 1 detik. Proses dekripsi untuk 7,2 MB dibutuhkan waktu 2,5 detik dibandingkan dengan 1,8 MB dibutuhkan waktu 0,4 detik.

Beberapa penelitian di atas memiliki persamaan dengan penelitian yang peneliti lakukan, yaitu sama-sama meneliti tentang penerapan algoritme kriptografi yang tepat untuk memenuhi unsur keamanan data pada perangkat IoT. Selain itu, beberapa penelitian terdahulu yang digunakan sebagai referensi memiliki kesamaan terhadap pemilihan algoritme yang digunakan. Perbedaan yang dilakukan pada penelitian ini berfokus pada implementasi algoritme AES 128 bit untuk keamanan data di perangkat mikrokontroler NodeMCU, khususya dalam enkripsi data yang akan dikirim dari mikrokontroler menuju *server* basis data. Setelah dilakukan proses implementasi algoritme tersebut, akan dilakukan pengujian fungsional, pengujian kinerja waktu dan mermori, serta pengujian keamanan.

2.2 Dasar Teori

2.2.1 Kriptografi

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi. Kata “seni” di dalam definisi di atas berasal dari fakta sejarah bahwa pada masa awal sejarah kriptografi, setiap orang mungkin mempunyai cara yang unik untuk merahasiakan pesan. Cara-cara unik tersebut mungkin berbeda-beda pada setiap pelaku kriptografi sehingga setiap cara menulis pesan rahasia itu mempunyai nilai estetika tersendiri sehingga kriptografi berkembang menjadi sebuah seni merahasiakan pesan (kata “*graphy*” di dalam “*cryptography*” itu sendiri sudah menyiratkan sebuah seni)[9].

2.2.2 *Advanced Encryption Standard* (AES) - Rijndael

Algoritme Rijndael merupakan algoritme yang ditetapkan oleh NIST sebagai AES pada bulan Oktober 2000. Algoritme Rijndael ditemukan oleh Vincent Rijmen dan Joan Daemen dari Belgia. Rijndael mendukung panjang kunci 128bit sampai 256bit dengan step 32bit. Panjang kunci dan ukuran blok dapat dipilih secara independen. Karena AES menetapkan bahwa ukuran blok harus 128bit, dan panjang kunci harus 128, 192, dan 256bit, maka dikenal AES-128, AES-192, AES-256. Setiap blok dienkripsi dalam jumlah putaran tertentu bergantung pada panjang kuncinya.

Tabel 2. 1 Jumlah putaran setiap blok pada AES

Varian AES	**Panjang Kunci (Nk *words*)**	**Ukuran Blok (Nb *words*)**	**Jumlah Putaran (Nr)**
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Catatan : 1 *word* = 32bit

Secara de-fakto, hanya ada dua varian AES, yaitu AES-128 dan AES 256, karena akan sangat jarang pengguna menggunakan kunci yang panjangnya 192bit.

Karena AES mempunyai panjang kunci paling sedikit 128bit, maka AES tahan terhadap serangan *exhaustive key search* dengan teknologi saat ini. Dengan panjang kunci 128bit, maka terdapat $2^{128} \sim 3{,}4 \times 10^{38}$ kemungkinan kunci. Jika digunakan sebuah mesin dengan semiliyar prosesor *parallel*, masing-masing dapat menghitung sebuah kunci setiap satu *pico* detik, maka akan dibutuhkan waktu 10^{10} tahun untuk mencoba seluruh kemungkinan kunci.

Garis besar algoritme Rijndael yang beroperasi blok 128-bit dengan kunci 128-bit adalah sebagai berikut:

1. *AddRoundKey*: melakukan XOR antara *state* awal (*plainteks*) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak Nr – 1 kali. Proses yang dilakukan pada setiap putaran adalah:

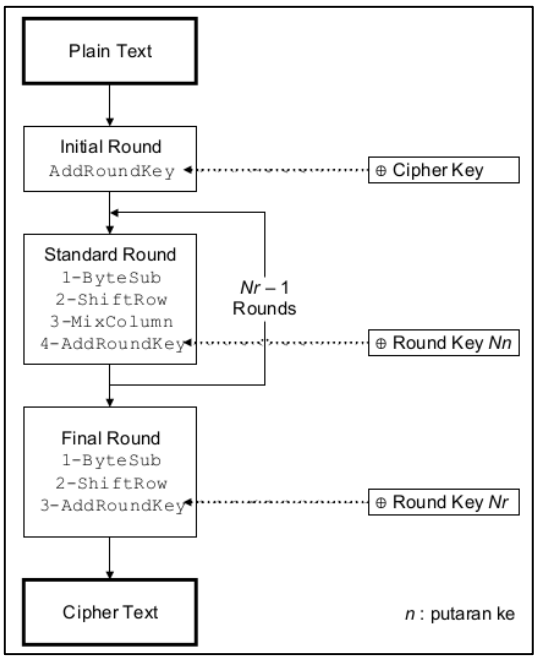
a. *ByteSub*: substitusi *byte* dengan menggunakan tabel substitusi (S- box). Tabel substitusi dapat dilihat pada tabel 1, sedangkan ilustrasi *ByteSub* dapat dilihat pada gambar 2.
b. *ShiftRow*: pergeseran baris-baris *array state* secara *wrapping*. Ilustarsi *ShiftRow* dapat dilihat pada gambar 3.
c. *MixColumn*: mengacak data di masing-masing kolom *array state*. Ilustarsi *MixColumn* dapat dilihat pada gambar 4.
d. *AddRoundKey*: melakukan XOR antara *state* sekarang dengan *round key*. Ilustarsi *AddRoundKey* dapat dilihat pada gambar 5.

3. *Final round*: proses untuk putaran terakhir:
   a. *ByteSub*.
   b. *ShiftRow*.
   c. *AddRoundKey*.

Diagram proses enkripsi AES dapat dilihat pada Gambar 1. Algoritme Rijndael mempunyai 3 parameter sebagai berikut:

a. *Plainteks* : *array* yang berukuran 16 *byte*, yang berisi data masukan.
b. *Cipherteks* : *array* yang berukuran 16 *byte*, yang berisi hasil enkripsi.
c. *Key* : *array* yang berukuran 16 *byte*, yang berisi kunci *ciphering* (disebut juga *cipher key*).

Dengan 16 *byte*, maka baik blok data dan kunci yang berukuran 128-bit dapat disimpan di dalam ketiga *array* tersebut (128 = 16 x 8). Selama kalkulasi *plainteks* menjadi *cipherteks*, status sekarang dari data disimpan di dalam *array* of *byte* dua dimensi, *state*, yang berukuran NROWS x NCOLS. Elemen *array state* diacu sebagai S[r,c], dengan $0 \leq r < 4$ dan $0 \leq c < Nc$ (Nc adalah panjang blok dibagi 32). Pada AES, Nc = 128/32 = 4.[15]

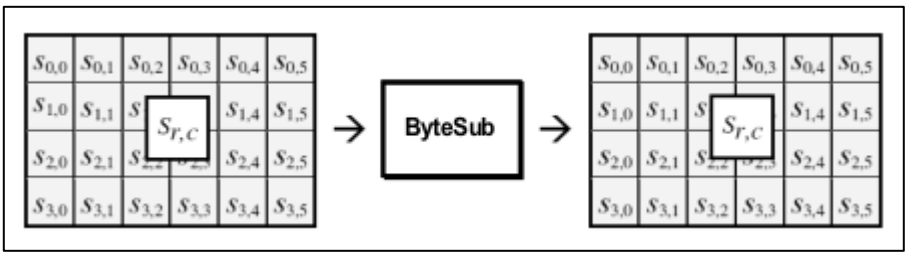


Gambar 2. 1 Diagram proses enrkipsi AES

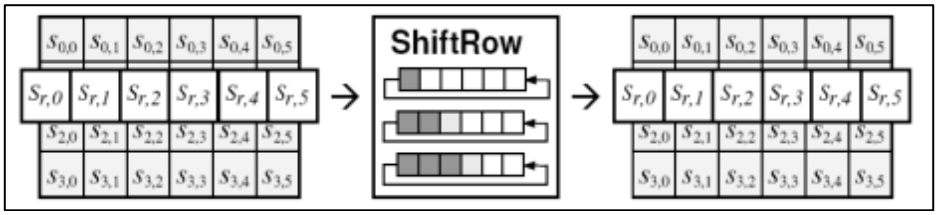
Tabel 2. 2 Tabel S-box yang digunakan dalam transofrmasi *ByteSub*() AES

hex	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

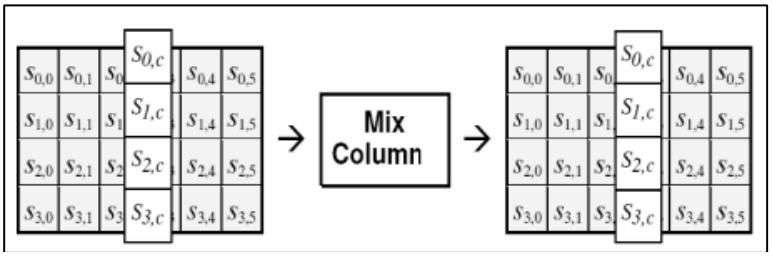
x



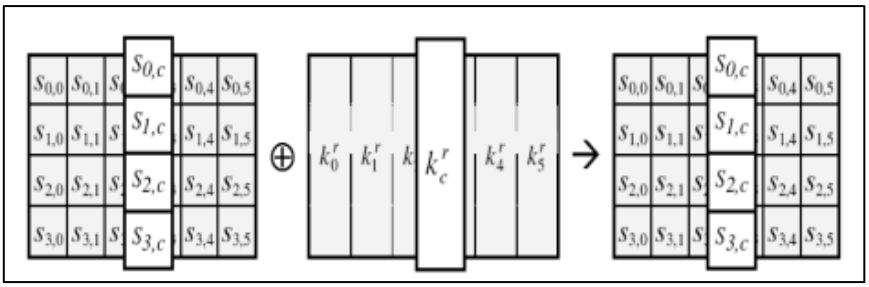
Gambar 2. 2 Ilustrasi Transformasi *ByteSub*() AES



Gambar 2. 3 Ilustrasi transformasi *ShiftRow*() AES



Gambar 2. 4 Ilustrasi transformasi *MixColumn*() AES



Gambar 2. 5 Ilustrasi transformasi *AddRoundKey*() AES

Pada penelitian ini algoritme Rijndael atau AES (*Advenced Encryption Standard*) akan digunakan pada *client* REST (*encryption*) dan *server* REST (*decryption*).

2.2.3 *Internet of Things*

Istilah *Internet of Things* (IoT) adalah sebuah konsep dimana suatu objek memiliki kemampuan untuk men-*transfer* data melalui jaringan tanpa memerlukan

interaksi manusia ke manusia atau manusia ke komputer. Konsep ini akan meningkatkan kapasitas data yang dihasilkan dan diolah pada setiap perangkat elektronik yang saling terhubung dalam jaringan. Pada era IoT, perangkat elektronik akan menggunakan *cloud* sebagai tempat penyimpanan data. Salah satu contoh penyedia layanan *cloud* adalah Amazon *Web Services* (AWS). Penggunaan *cloud* bertujuan supaya data dapat diakses oleh beberapa perangkat sekaligus kapanpun dan dimanapun. Selain itu, banyaknya data yang terdapat pada era IoT tidak memungkinkan data untuk disimpan pada penyimpanan internal.

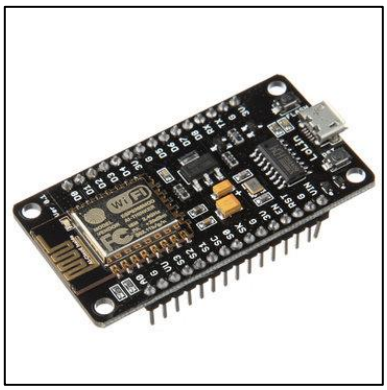
2.2.4 Mikrokontroler NodeMCU 1.0

NodeMCU 1.0 merupakan papan pengembangan mikrokontroler berbasis Arduino yang menggunakan modul 802.11 bertipe ESP12-E. NodeMCU 1.0 sangat kompatibel dengan bahasa pemrograman C dan komunikasi antarperangkat atau internet secara nirkabel. NodeMCU memiliki dimensi sebesar 47mm x 31mm. Selain itu, NodeMCU termasuk mikrokontroler yang memiliki prosesor dan memori kecil. Spesifikasi NodeMCU dapat dilihat pada Tabel 2.1.

Tabel 2. 3 Spesifikasi perangkat NodeMCU

Tegangan operasi	5 *volt*
Digital I/O (*Input/Output*) *pin*	30 buah
Clock Speed	80 MHz
SRAM (*Static Random Access Memory*)	8KB
Memori dinamis	81920 *byte*
Storage	4 MB

- Memori dinamis merupakan memori virtual yang dialokasikan dari *storage* berfungsi untuk membantu penyimpanan sementara SRAM ketika hampir penuh.
- *Storage* atau *flash memory* merupakan penyimpanan yang digunakan untuk menyimpan program.



Gambar 2. 6 Mikrokontroler NodeMCU 1.0

Pada penelitian ini digunakan mikrokontroler NodeMCU untuk membangun *client* REST.

2.2.5 *Web Service*

Web Service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung *interaction and interoperability* antar sistem pada suatu jaringan. *Web service* digunakan sebagai suatu fasilitas yang menyediakan layanan (dalam bentuk informasi atau data) kepada sistem lain, sehingga dapat berinteraksi dengan sistem tersebut melalui layanan-layanan yang disediakan. *Web service* menyimpan data informasi dalam format JSON atau XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda *platform*, sistem operasi, dan bahasa pemrograman[19].

2.2.6 REST

REST (*Representational State Transfer*) merupakan standar arsitektur komunikasi berbasis *web* yang sering diterapkan dalam pengembangan layanan berbasis *web*. Umumnya menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai *protocol* untuk komunikasi data[20] Pada arsitektur REST, REST *server* menyediakan *resources* (sumber daya/data) dan REST *client* mengakses dan menampilkan *resource* tersebut untuk penggunaan selanjutnya. Setiap *resource* diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. *Resource* tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

Berikut metode HTTP yang umum digunakan dalam arsitektur berbasis REST :

1. GET

   Metode GET menyediakan hanya akses baca pada *resource*.

2. PUT

   Metode PUT digunakan untuk menciptakan *resource* baru.

3. DELETE

   Metode DELETE digunakan untuk menghapus *resource*.

4. POST

   Metode POST digunakan untuk memperbarui *resource* yang ada atau membuat *resource* baru.

5. OPTIONS

   Metode OPTIONS digunakan untuk mendapatkan operasi yang di *support* pada *resource*.

Web service adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data biasa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada *platform* yang berbeda. Contoh implementasi dari *web service* antara lain adalah SOAP dan REST.

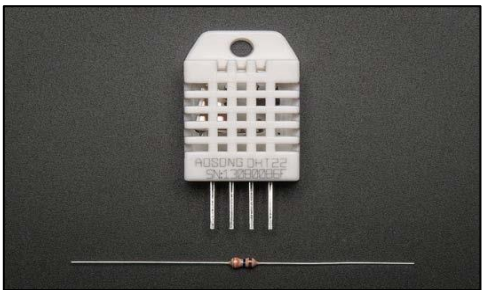
2.2.7 Flask

Flask adalah *micro web framework* yang ditulis dalam bahasa pemrograman Python dan berdasarkan *Werkzeug toolkit* dan *template engine* Jinja2. Flask disebut *micro framework* karena tidak membutuhkan alat-alat tertentu atau pustaka. Flask tidak memiliki database *abstraction layer*, validasi *form*, atau komponen lain di mana sudah ada pustaka pihak ketiga yang menyediakan fungsi umum. Namun, Flask mendukung ekstensi yang dapat menambahkan fitur aplikasi seolah-olah mereka diimplementasikan dalam Flask itu sendiri. Ekstensi yang ada untuk *object-relational mapper*, validasi *form*, penanganan unggahan, berbagai teknologi otentikasi terbuka, dan beberapa 12 alat-alat yang terkait kerangka umum[12].

Pada penelitian ini digunakan *framework* Flask untuk membangun *web service* REST.

2.2.8 DHT-22

DHT - 22 (juga disebut sebagai AM2302) adalah sensor digital yang dapat mengukur suhu dan kelembaban udara di sekitarnya. Memiliki tingkat stabilitas yang sangat baik serta fitur kalibrasi yang sangat akurat. Koefisien kalibrasi disimpan dalam OTP program *memory*, sehingga ketika internal sensor mendeteksi sesuatu, maka *module* ini menyertakan koefisien tersebut dalam kalkulasinya. Ilustrasi dari sensor DHT-22 diperlihatkan pada gambar 2.7.

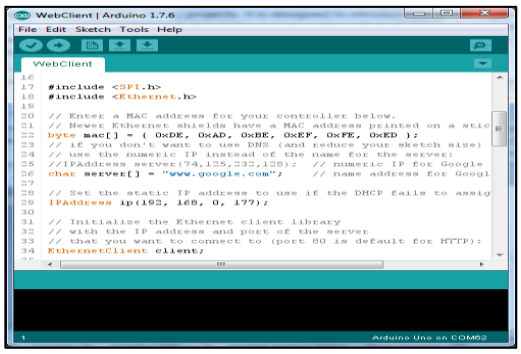


Gambar 2. 7 Sensor DHT-22

Pada penelitian ini digunakan sensor DHT-22 untuk melakukan sensing suhu dan kelembaban sehingga nantinya digunakan sebagai data untuk implementasi dan pengujian.

2.2.9 Arduino IDE

Arduino IDE merupakan lingkungan pengembangan yang dibuat dengan bahasa Java dan berasal dari *Processing* IDE. Program atau kode yang ditulis untuk papan Arduino dinamakan *sketch*. Arduino IDE sudah dilengkapi *file* pustaka tambahan yang berisi fungsi/*method* seperti menghubungkan ke jaringan dengan Wifi/*Ethernet*, membuat *server* sederhana, mengendalikan motor *stepper*, komunikasi data seri, dan sebagainya[11]. Ilustrasi *interface* dari Arduino IDE diperlihatkan pada gambar 2.8.



Gambar 2. 8 Arduino IDE

1. *Sketch*

Sketch adalah nama yang digunakan Arduino untuk suatu program. Ini adalah unit kode yang diunggah dan dijalankan di papan Arduino.[1]

2. Setup() dan Loop()

Ada dua fungsi khusus yang merupakan bagian dari setiap *sketch* Arduino: setup() dan loop(). Setup() dipanggil sekali, saat *sketch* dimulai. Ini adalah tempat yang baik untuk melakukan tugas pengaturan seperti mengatur mode *pin* atau menginisialisasi *library*. Fungsi loop() dipanggil berulang kali dan merupakan jantung dari sebagian besar sketsa.[1]

3. *Time*

a. Delay()

Menjeda program untuk jumlah waktu (dalam milidetik) yang ditentukan sebagai parameter. (Ada 1.000 milidetik dalam satu detik.)

b. DelayMicroseconds()

Menjeda program untuk jumlah waktu (dalam mikrodetik) yang ditentukan oleh parameter. Ada seribu mikrodetik dalam milidetik dan sejuta mikrodetik dalam satu detik.

c. Micros()

Menghitung waktu(mikrodetik) yang dibutuhkan dalam mengeksekusi kode program pada perangkat mulai dari awal hingga akhir.

Pada penelitian ini menggunakan Arduino IDE untuk melakukan pengembangan sistem pada sisi *client*. Arduino IDE memiliki fitur yang beragam dan menyediakan *library* untuk kebutuhan pengembangan.

2.2.10 SQLite

SQLite adalah sebuah *open source* dengan *database* relasional. Awalnya dirilis pada tahun 2000, SQLite dirancang untuk menyediakan cara yang nyaman untuk aplikasi untuk mengelola data tanpa *overhead* yang sering muncul dengan sistem manajemen *database* relasional khusus. SQLite memiliki reputasi baik karena sangat *portabel*, mudah digunakan, kompak, efisien, dan dapat diandalkan.

Pada penelitian ini menggunakan SQLite sebagai *database* untuk menyimpan data suhu dan kelembaan. SQLite dipilih karena sederhana dan tidak memerlukan instalasi dan konfigurari yang rumit. Selain itu, SQLite tersedia sebagai pustaka untuk Flask.

2.2.11 Wireshark

Wireshark adalah *tool* yang di tujukan untuk penganalisisan paket data jaringan. Wireshark melakukan pengawasan paket secara waktu nyata (*real time*) dan kemudian menangkap data dan menampilkannya selengkap mungkin. Wireshark bisa digunakan secara gratis karena aplikasi ini berbasis sumber terbuka. Aplikasi Wireshark dapat berjalan di banyak *platform*, seperti Linux, Windows, dan Mac[13]. Struktur dari Wireshark *graphical user interface* adalah sebagai berikut :

1. *Command menu*

   Command menu merupakan daftar yang dibutuhkan pada Wireshark.

2. *Display filter specification*

   Display filter specification berfungsi untuk mem-*filter* paket data.

3. *Listing of captured packets*

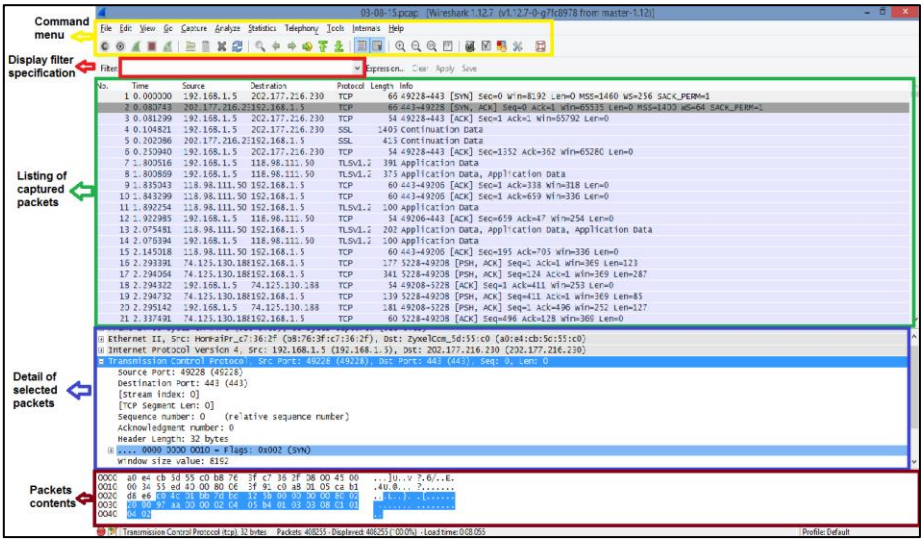
   Listing of captured packets merupakan paket data yang tertangkap oleh Wireshark.

4. *Details of selected packet header*

   Details of selected packet header data lengkap tentang *header* dari suatu paket.

5. *Packet contents*

Packet contents isi dari suatu paket data.

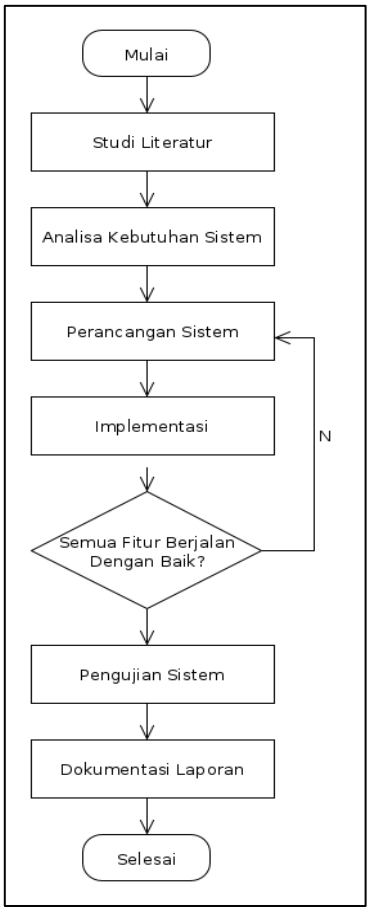


Gambar 2. 9 Struktur Wireshark

BAB III
METODOLOGI PENELITIAN

3.1 Diagram Alir Penelitian

Diagram alir penelitian laporan implementasi algoritme AES 128bit pada mikrokontroler NodeMCU menggunakan arsitektur *web service* REST untuk keamanan pengiriman data, dapat dilihat pada gambar 3.1.



Gambar 3. 1 Diagram Alir Penelitian

Penjelasan mengenai masing-masing proses pada diagram alir penelitian Gambar 3.1 di atas adalah sebagai berikut:

1. Pada tahap studi literatur akan dilakukan pengumpulan literatur yang berkaitan dengan implementasi algoritme AES untuk keamanan data pada perangkat IoT, Mikrokontroler NodeMCU, *web service* REST, serta alat-alat yang dibutuhkan.

2. Pada tahap analisa kebutuhan sistem akan dilakukan analisa kebutuhan untuk perangkat keras dan perangkat lunak untuk membangun sistem pada sisi *client* dan *server* REST.
3. Pada tahap perancangan sistem akan dilakukan perancangan perangkat keras, perangkat lunak, perancangan *database* dan perancangan algoritme. Pada perancangan perangkat keras meliputi perancangan proses komunikasi *client* dan *server* REST. Pada tahap perancangan perangkat meliputi perancangan alur kerja *client* dan *server* REST dalam bentuk diagram. Pada tahap perancangan *database* akan dilakukan perancangan model *database* yang diperlukan sesuai dengan kebutuhan sistem. Pada tahap perancangan algoritme akan dilakukan perancangan berupa ilustrasi dari peroses enkripsi dan dekripsi algoritme AES.
4. Pada tahap implementasi akan diimplementasikan sesuai dengan perancangan sistem untuk sisi *client* dan *server* REST. Pada sisi *client*, dimulai dengan penyusunan perangkat keras, memprogram dan mengimplementasikan algoritme AES 128bit untuk enkripsi data. Pada sisi *server*, dibangun sebuah REST *Server* menggunakan *framework* Flask yang berfungsi sebagai jalur komunikasi antar *client* dan *server database*. Algroitma AES128bit diimplementasikan pada REST *server* untuk melakukan dekripsi data yang diterima oleh *client* sebelum disimpan ke dalam *database*.
5. Jika semua fitur berjalan dengan baik, maka akan dilanjutkan ke tahap pengujian sistem. Jika sistem belum berjalan dengan baik, maka akan dilakukan perbaikan dari tahap perancangan sistem.
6. Pada tahap pengujian sistem, akan dilakukan pengujian terhadap sistem yang telah dibangun dengan skenario uji meliputi fungsional, pengujian kinerja waktu dan memori serta pengujian keamanan.
7. Pada tahap dokumentasi dan laporan akan dilakukan pecatatan dari hasil pengujian.

3.2 Analisis Kebutuhan Sistem

3.2.1 Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan dalam pengimplementasian algoritme AES 128bit dan pengembangan sistem dalam penelitian ini dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Kebutuhan perangkat keras sistem

Perangkat Keras	Keterangan
Sensor DHT-22	Modul sensor untuk melakukan *sensing* suhu dan kelembaban.
NodeMCU 1.0 (ESP-12E *Module*)	Perangkat yang digunakan sebagai mikrokontroler bagi sensor DHT-22. Selain itu, NodeMCU juga bertindak sebagai *client* REST pada sistem.
Laptop ASUS	Perangkat laptop ini bertindak sebagai *server* basis data dan *server* REST
Wi-fi *Router*	Wi-fi *router* untuk memberikan koneksi jaringan nirkabel pada NodeMCU dan Laptop ASUS.

3.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan dalam pengimplementasian algoritme AES 128bit dan pengembangan sistem dalam penelitian ini dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Kebutuhan perangkat lunak sistem

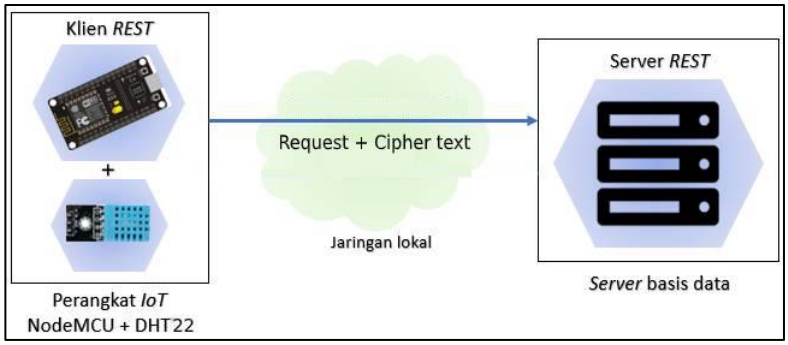
Perangkat Lunak	Keterangan
Linux Ubuntu 18.04 64bit	Sistem operasi yang digunakan untuk menjalankan IDE Arduino dan *Web Server* REST
IDE Arduino	IDE yang digunakan untuk memprogram dan mengimplementasi kan algoritme AES 128bit dan REST

	Client pada mikrokontroler NodeMCU
Visual Studio Code (VSCode)	*Code Editor* yang digunakan untuk membangun *Web Server* REST
Microframework Flask	*Framework* yang digunakan untuk membangaun *Web Server* REST
Wireshark	*Tools* yang digunakan untuk melakukan *sniffing* pada proses pengiriman data dari *client* ke *server*.

3.3 Perancangan Sistem

3.3.1 Perancangan Perangkat Keras

Pada perancangan perangkat keras meliputi perancangan proses komunikasi antar sisi *client* REST dan *server* REST dapat dilihat pada Gambar 3.2.

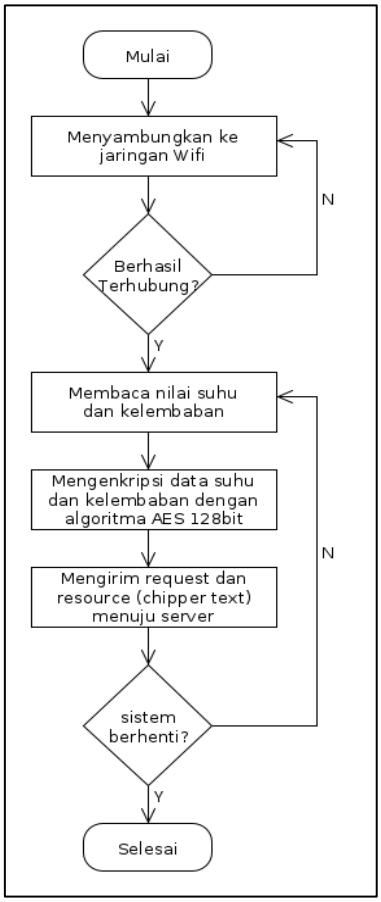


Gambar 3. 2 Rancangan proses komunikasi *client-server* REST

Gambar 3.2 menjelaskan tentang rancangan proses komunikasi antarperangkat yang ada pada sistem. Secara umum terlihat bahwa sistem menerapkan arsitektur komunikasi *web service* REST. Terdapat unsur pembentuk dalam arsitektur *web service* yaitu *client* dan *server*. Sistem akan menggunakan jaringan *local* dalam proses komunikasi antara *client* dengan *server*. *Client* REST akan mengirimkan *request* POST beserta data suhu dan kelembaban yang sudah dienkripsi menggunakan algoritme AES 128bit menjadi *cipher text* kepada *server* REST. *Server* bertugas menerima *request* dan data dari *client* tersebut, kemudian dilakukan dekripsi data dan menyimpan data tersebut ke *server* basis data.

3.3.2 Perancangan Perangkat Lunak

Pada perancangan perangkat lunak meliputi perancangan alur kerja *client* REST dan *server* REST dapat dilihat pada Gambar 3.3 dan Gambar 3.4.

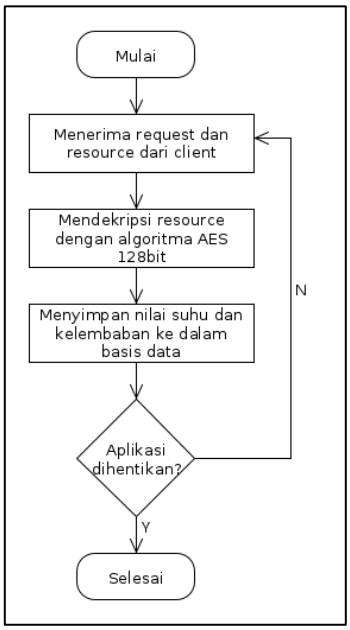


Gambar 3. 3 Rancangan alur kerja *client* REST

Penjelasan mengenai masing-masing proses pada diagram alir Gambar 3.3 di atas adalah sebagai berikut:

1. Pada tahap menyambungkan ke jaringan Wifi, sistem dirancang dapat tersambung secara otomatis ke jaringan Wifi.
2. Jika sistem berhasil terhubung, maka akan dilanjutkan ke tahap selanjutnya. Jika sistem tidak berhasil terhubung, maka akan dilakukan perulangan peroses dari tahap menyambungkan ke jaringan Wifi.
3. Pada tahap membaca nilai suhu dan kelembaban akan dilakukan pembacaan terhadap nilai suhu dan kelembaban dari sensor DHT-22.

4. Pada tahap mengenkripsi data suhu dan kelembaban dengan algoritme AES 128bit, akan dilakukan enkripsi menggunakan algoritme AES 128bit kemudian didapatkan *cipher text*.
5. Pada tahap mengirim *request* dan *resource* (*cipher text*) menuju *server*, data *cipher text* akan dikirim menuju *server* REST menggunakan *method* POST.
6. Jika sistem dihentikan atau dalam keadaan *off*, maka rangkaian proses pada sistem akan berhenti. Jika program tidak berhenti, maka akan dilakukan perulangan peroses dari tahap membaca nilai suhu dan kelembaban.



Gambar 3. 4 Rancangan alur kerja *server* REST

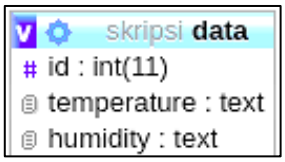
Penjelasan mengenai masing-masing proses pada diagram alir Gambar 3.4 di atas adalah sebagai berikut:

1. Pada tahap menerima *request* dan *resource* dari *client*, *server* akan menerima *request* dan *resource* (*cipher text*) dari *client.*
2. Pada tahap mendekripsi *resource* dengan algoritme AES 128bit, akan dilakukan dekripsi data *cipher text* menggunakan algoritme AES 128bit menjadi *plain text* sehingga didapatkan data semula dalam bentuk plainteks.
3. Pada tahap menyimpan data suhu dan kelembaban ke dalam basis data, data suhu dan kelembaban yang didapatkan kemudian disimpan ke dalam *database*.

4. Jika sistem dihentikan atau *server* berhenti, maka rangkaian proses pada sistem akan berhenti. Jika sistem tidak dihentikan, maka sistem akan menunggu *request* dari *client.*

3.3.3 Perancangan *Database*

Pada Rancangan *database* akan dibuat model *database* yang diperlukan untuk menyimpan data suhu dan kelembaban yang dikirim dari *client* REST. Rancangan *database* memiliki 1 entitas dengan 3 atribut dapat dilihat pada Gambar 3.6.



Gambar 3. 5 Rancangan *Database*

Gambar 3.5 menjelaskan rancangan *database* hanya memiliki 1 entitas yakni data yang memuat atribut id, *temperature*, dan *humidity*. Id merupakan atribut yang dibuat AUTOINCREMENT untuk setiap data baru yang masuk. *Temperature* mewakili atribut untuk menyimpan data suhu dan *humidity* mewakili atribut untuk menyimpan data kelembaban.

3.3.4 Perancangan Algoritme

1. Representasi Data

AES merepresentasikan data dengan cara urutan *byte* dan *bit* (0 atau 1 pada b_n), dimana data diturunkan dari urutan *input* 128 *bit* per blok. *Bit-bit* tersebut diberi indeks mulai dari 0 sampai dengan 127 ($0 \leq i < 128$). Setiap urutan 8 *bit* (1 *byte*) diberlakukan sebagai entitas tunggal yang merupakan elemen *finite field* dengan representasi polinomial pada persamaan berikut.

$$b(x) = b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

Pengindeks-an *bit* dalam *byte* pada blok dapat dilihat pada Gambar 3.6.

Urutan *Bit*	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
Input	*Input*$_0$								*Input*$_1$								…
Posisi *Bit*/*Byte*	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…

State	$State_0$	$State_1$	…

Gambar 3. 6 Pengindeks-an *bit* dalam *byte* pada blok

Cipher AES dilakukan pada *array byte* 2 dimensi yang disebut *state.* Blok data disusun dalam *state* yang terdiri atas empat baris Nb *byte* (Nb = panjang blok/4 adalah 4 untuk AES 128). Setiap *byte* diberi dua indeks yang menyatakan posisinya, dinyatakan sebagai $s_{r,c}$ atau s[r,c], dengan indeks baris r (*row*) dalam *interval* $0 \leq r < 4$, sedangkan indeks kolom c (*coloum*) dalam $0 \leq c < Nb$. Data dalam *state* menginformasikan hasil setiap tahap transformasi (*intermediate result*).

Input dikopi ke *state array* pada permulaan *cipher* dan *inverse cipher*, kemudian *state* diperbaharui pada akhir setiap transformasi. Nilai *state* pada transformasi yang terakhir kemudian dikopi ke *output* kembali (lihat Gambar 3.3.) dengan pengindeks-an yang serupa Tabel 3.6.

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

→

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

→

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Gambar 3. 7 *State array* pada *input* dan *output*

State juga dapat dipandang sebagai 4 *word byte*, dengan indeks baris r dari $s_{r,c}$ menyatakan indeks dari keempat *byte* dalam setiap *word* . Dengan kata lain, *state* ekivalen dengan *array* dari empat word yang berindeks c (indek kolom dari s r,c) seperti dilihat pada Gambar 3.8.

$w_0 = s_{0,0} \, . \, s_{1,0} \, . \, s_{2,0} \, . \, s_{3,0}$

$w_1 = s_{0,1} \, . \, s_{1,1} \, . \, s_{2,1} \, . \, s_{3,1}$

$w_2 = s_{0,2} \, . \, s_{1,2} \, . \, s_{2,2} \, . \, s_{3,2}$

$w_3 = s_{0,3} \, . \, s_{1,3} \, . \, s_{2,3} \, . \, s_{3,3}$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

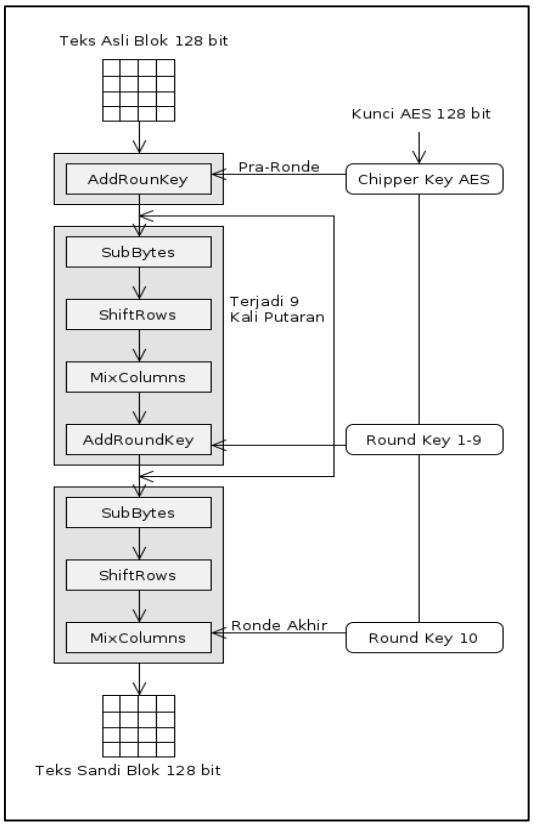
=

w_0	w_1	w_2	w_3

Gambar 3. 8 *State array* ekivalen pada *word array*

2. Enkripsi

Pada proses enkripsi dilakukan menggunakan empat fungsi dasar pada algoritme AES yang meliputi *SubBytes, ShiftRows,MixColumns,* dan *AddRoundKey*. Rentetan fungsi tersebut dijalankan sebanyak Nr-1 sebagai *loop* utama, setiap *loop* disebut *round* (Nr = 10 *round* untuk AES 128). *AddRoundKey* dieksekusi sebagai *round* inisial sebelum *loop* utama. Setelah *loop* utama berakhir (Sembilan *round*), *SubBytes, ShiftRows,* dan *AddRoundKey,* dieksekusi secara berturut-turut sebagai *final round*. Ilustrasi proses enkripsi AES diperlihatkan pada Gambar 3.9.



Gambar 3. 9 Ilustrasi proses enkripsi AES

Penjelasan mengenai masing-masing proses pada Gambar 3.9 di atas adalah sebagai berikut:

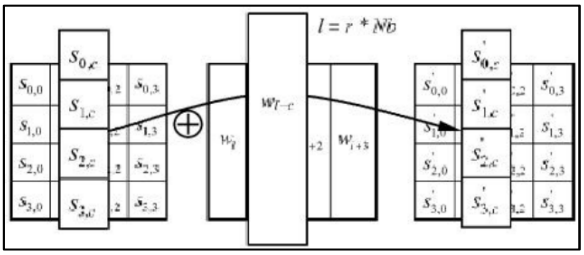
a. *AddRoundKey*

Transformasi ini melakukan operasi XOR terhadap sebuah *round key* dengan *array state* yang baru. Nilai awal dari *round key* adalah w_0, untuk *round* = 0, ditambahkan pertama kali pada *cryptographic round*. Kemudian

setiap *round* untuk 1 ≤ *round* ≤ Nr, kemudian 32-bit yang berbeda pada *round key* w_i ditambahkan.

$[S'0,c , S'1,c , S'2,c , S'3,c] = [S0,c , S1,c , S2,c , S3,c] \oplus [w_{round} * Nb+c]$

Untuk $0 \leq c < Nb$.

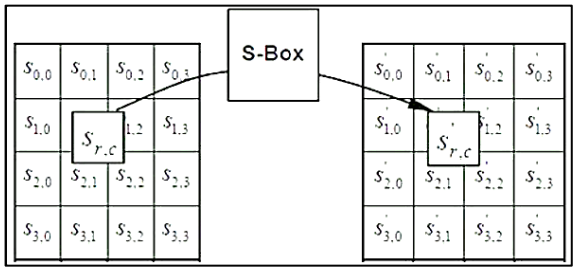


Gambar 3. 10 *AddRoundKey*[5]

b. *SubBytes*

Transformasi ini merupakan suatu operasi substitusi *nonlinier* yang beroperasi secara mandiri pada setiap *byte* dengan menggunakan tabel substitusi S-Box. Cara pensubstitusian adalah sebagai berikut: untuk setiap *byte array state,* misalkan S[r,c] = xy, yang dalam hal ini xy adalah digit heksadesimal dari nilai S[r,c], maka nilai substitusinya dinyatakan dengan S'[r,c] adalah elemen di dalam S-box yang merupakan perpotongan baris x dengan kolom y.

Misalnya S[0,0] = 19, maka S'[0.0] = d4.

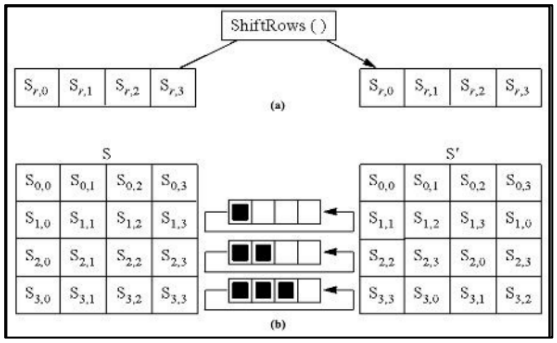


Gambar 3. 11 *SubBytes*[5]

c. *ShiftRows*

Transformasi ini merupakan langkah permutasi yang dieksekusi melalui pergeseran secara *wrapping* (siklis) pad 3 baris terakhir dari *aray state*. Jumlah pergeseran bergantung pada nilai baris (r). Baris r = 1 digeser

sejauh 1 *byte*, baris r = 2 digeser sejauh 2 *byte*, dan baris r = 3 digeser sejauh 3 *byte*. Baris r = 0 tidak digeser.



Gambar 3. 12 *ShitRows*[5]

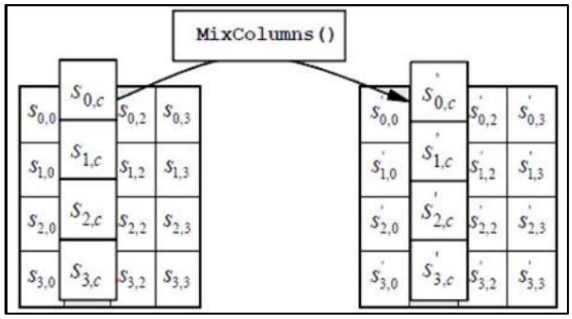
d. *MixColumns*

Transformasi ini merupakan pengoperasian *state* kolom demi kolom. Operasi ini dilakukan pada *state* kolom, dengan mengkonversikan setiap kolom sebagai *polynomial*. Kolom dianggap sebagai *polynomial* pada GF(28). Tranformasi ini dapat digambarkan pada Gambar 3.14 dengan perkalian matriks seperti persamaan pada Gambar 3.13.

$$S'_{0,c} = ([02]\bullet S_{0,c}) \oplus ([03]\bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$
$$S'_{1,c} = S_{0,c} \oplus ([02]\bullet S_{1,c}) \oplus ([03]\bullet S_{2,c}) \oplus S_{3,c}$$
$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus ([02]\bullet S_{2,c}) \oplus ([03]\bullet S_{3,c})$$
$$S'_{3,c} = ([03]\bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus ([02]\bullet S_{3,c})$$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} x \begin{bmatrix} S\,0,c \\ S\,1,c \\ S\,2,c \\ S\,3,c \end{bmatrix} = \begin{bmatrix} S'\,0,c \\ S'\,1,c \\ S'\,2,c \\ S'\,3,c \end{bmatrix}$$

Gambar 3. 13 Persamaan matriks *MixColumns*[5]



Gambar 3. 14 *MixColumns*[5]

3. Ekspansi Kunci

Algoritme AES membuat suatu ekspansi kunci dari *cipher key* untuk menghasilkan *key schedule* yang digunakan untuk proses enkripsi. Kunci direpresentasikan menjadi *word* (*w*[i]) lihat Gambar 3.8. serupa dengan *state*, akan tetapi elemen *state*-nya adalah *cipher key*. Ekspansi kunci yang diperlukan AES Nb(Nr+1) *word*, sehingga untuk AES-128 membutuhkan 4(10+1) *word* = 44 *word*.

Beberapa langkah yang ditempuh untuk membuat *key schedule* yaitu *Rotword*(), *SubWord*(), dan *Rcon*().

RotWord() adalah Jika w [i] direpresentasikan dengan *array* baris atau kolom menjadi baris (*transpose*), maka dapat di ilustrasikan dengan menggeser sekali ke kiri pada posisi *byte* seperti yang dilakukan *shiftrows*() pada baris kedua. Misal w[i] = (a_0,a_1,a_2,a_3), maka didapat *RotWord*(w[i]) = (a_1,a_2,a_3,a_0).

SubWord() yaitu subtitusikan setiap *byte* yang dikonversikan kebentuk heksadesimal dengan tabel S-Box seperti yang dilakukan *SubBytes*(). Misal w[i] = CF4F3C09 , dengan mensubtitusikan ketabel S-Box menghasilkan *SubWord*(w[i]) = 8A84EB01, dimana CF menjadi 8A, 4F menjadi 84, 3C menjadi EB, dan 09 menjadi 01.

Rcon[i] merupakan suatu komponen tetap (konstanta) *word* dari *round* dalam perhitungan ekspansi ke dalam *key schedule*. Adapun nilainya untuk AES-128 yang menggunakan 10 kali putaran dari persamaan yang diperlihatkan pada Gambar 3.15.

Rcon[i] = [x_i , ‘00’, ‘00’, ‘00’]

Gambar 3. 15 Persamaan *Rcon*

Rcon[1] = [x^0 , ‘00’, ‘00’, ‘00’] = [‘01’, ‘00’, ‘00’, ‘00’] = 01000000

Rcon[2] = [x^1 , ‘00’, ‘00’, ‘00’] = [‘02’, ‘00’, ‘00’, ‘00’] = 02000000

Rcon[3] = [x^2 , ‘00’, ‘00’, ‘00’] = [‘04’, ‘00’, ‘00’, ‘00’] = 04000000

Rcon[4] = [x^3 , ‘00’, ‘00’, ‘00’] = [‘08’, ‘00’, ‘00’, ‘00’] = 08000000

Rcon[5] = [x^4, ‘00’, ‘00’, ‘00’] = [‘10’, ‘00’, ‘00’, ‘00’] = 10000000

Rcon[6] = [x^5 , ‘00’, ‘00’, ‘00’] = [‘20’, ‘00’, ‘00’, ‘00’] = 20000000

Rcon[7] = [x^6 , ‘00’, ‘00’, ‘00’] = [‘40’, ‘00’, ‘00’, ‘00’] = 40000000

Rcon[8] = [x^7, ‘00’, ‘00’, ‘00’] = [‘80’, ‘00’, ‘00’, ‘00’] = 80000000

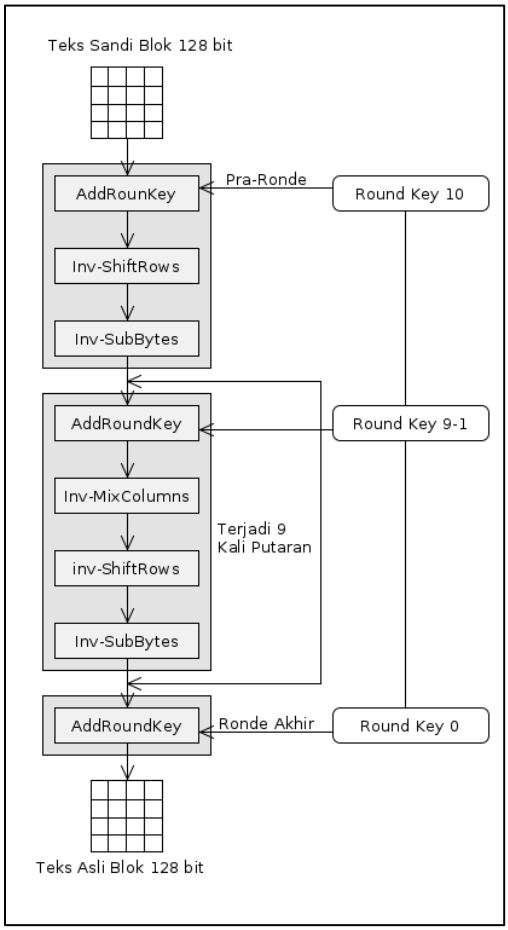
$$Rcon[9] = [x^8, '00', '00', '00'] = [x^7 \bullet x, '00', '00', '00'] = 1B000000$$

$$X^7 \bullet x = xtime(x^7) = xtime(80) = [leftshift(80)] = '1B'$$

$$Rcon[10] = [x^9, '00', '00', '00'] = [x^8 \bullet x, '00', '00', '00'] = 36000000$$

4. Dekripsi

Pada tahap dekripsi AES menggunakan transformasi invers dari smua transformasi dasar yang digunakan pada tahap enkripsi. Setiap transformasi dasar dari algoritme kriptografi AES memiliki transformasi *invers*, yaitu *InvSubBytes*, *InvShiftRows* dan *InvMixColumns*. *AddroundKey* merupakan transformasi yang bersifat *self-invers* dengan syarat menggunakan kunci yang sama. Ilustrasi proses dekripsi AES diperlihatkan pada Gambar 3.16.



Gambar 3. 16 Ilustrasi proses dekripsi AES

Penjelasan mengenai masing-masing proses pada Gambar 3.16 di atas adalah sebagai berikut:

a. *InvSubBytes*

InvSubBytes() perubahan hanya pada tabel S-box yang digunakan yaitu S-Box^{-1} . *Invers* dari tabel S-Box yang digunakan untuk *invers SubBytes*() tersedia sebagai S-Box -1 seperti Tabel 3.3

Tabel 3. 3 Tabel S-Box^{-1} untuk transformasi *invers SubBytes*

	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

b. *InvShiftRows*

Kebalikan *ShiftRows*() ini berlangsung dengan menggeser siklik kea rah berlawanan. Baris ke dua digeser siklik ke kanan skali, baris ke tiga dua kali, baris ke empat tiga kali.

c. *InvMixColumns*

Operasi *state* per kolm diwujudkan *MixColumns* memiliki kebalikan berupa persamaan yang diperlihatkan pada Gambar

$$
\left.
\begin{aligned}
s'_{0,c} &= ([0E] \bullet s_{0,c}) \oplus ([0B] \bullet s_{1,c}) \oplus ([0D] \bullet s_{2,c}) \oplus ([09] \bullet s_{3,c}) \\
s'_{1,c} &= ([09] \bullet s_{0,c}) \oplus ([0E] \bullet s_{1,c}) \oplus ([0B] \bullet s_{2,c}) \oplus ([0D] \bullet s_{3,c}) \\
s'_{2,c} &= ([0D] \bullet s_{0,c}) \oplus ([09] \bullet s_{1,c}) \oplus ([0E] \bullet s_{2,c}) \oplus ([0B] \bullet s_{3,c}) \\
s'_{3,c} &= ([0B] \bullet s_{0,c}) \oplus ([0D] \bullet s_{1,c}) \oplus ([09] \bullet s_{2,c}) \oplus ([0E] \bullet s_{3,c})
\end{aligned}
\right\}
$$

$$
\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} = \begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix}
$$

Gambar 3. 17 Persamaan matriks *MixColumns*

3.4 Implementasi Sistem

Setelah tahap perancangan akan dilakukan proses implementasi. Terdapat empat tahap dalam implementasi, yakni penyusunan perangkat, pembangunan *client* REST, pembangunan *server* REST dan *database*.

1. Penyusunan Perangkat

Pada tahap penyusunan perangkat, mikrokontroler NodeMCU dan sensor DHT-22 akan dihubungkan menggunakan kabel *jumper*. Proses penyusunan perangkat akan dilakukan sesuai dengan rancangan perangkat keras pada tahap perancangan perangkat keras.

2. Pembangunan *Client* REST

Pada tahap pembangunan *client* REST, akan diimplementasikan rancangan perangkat lunak pada sisi *client* ke dalam mikrokontroler NodeMCU dengan menggunakan bahasa pemrograman C. Arduino IDE akan digunakan sebagai alat bantu dalam proses implementasi.

3. Pembangunan *Server* REST

Pada tahap pembangunan *server* REST, akan diimplementasikan rancangan perangkat lunak pada sisi *server* menggunakan bahasa pemrograman Python. Flask akan digunakan sebagai *framework* dalam membangun *web service* REST.

4. Pembangunan *Database*

Pada tahap pembangunan *database*, rancangan *database* akan diimplementasikan menggunakan SQLite. SQLite merupakan sebuah *library* yang sudah terdapat pada *framework* Flask.

3.5 Pengujian Sistem

Pada tahap pengujian sistem, akan dilakukan pengujian terhadap sistem yang telah dibangun. Pengujian dilakukan dengan tujuan untuk mengetahui fungsional, kinerja waktu dan memori serta pengujian keamanan. Berikut merupakan penjelasan dari masing-masing pengujian.

1. Pengujian Fungsional

Pengujian Funsionalitas dilakukan untuk melihat kesesuaian fungsi-fungsi hasil implementasi dengan perancangan. Fungsi-fungsi tersebut meliputi fungsi menyambungkan ke jaringan wifi, membaca nilai suhu dan kelembaban, membungkus data suhu dan kelembaban ke dalam JSON, mengubah JSON menjadi bentuk biner, mengenkripsi *plain text* menjadi *cipher text*, mengirim HTTP *request* dan *cipher text* menuju *server*, menerima HTTP *request* dan *cipher text* dari *client*, mengubah kumpulan bit biner menjadi bentuk ASCII dan menyimpan data ke dalam basis data. Hasil pengujian yang diharapkan nantinya adalah masing-masing fungsi yang dirancang berhasil dijalankan dengan baik.

2. Pengujian Kinerja Waktu

Pengujian kinerja waktu dilakukan untuk mengetahui waktu yang dibutuhkan dalam proses enkripsi data pada prangkat mikrokontroler NodeMCU menggunakan fungsi micros(). Fungsi ini ditempatkan pada fungsi loop() utama dari kode program (*sketch*). Fungsi loop() merupakan fungsi utama dan secara *default* harus ada dalam kode program. Sekenario dilakukan sebanyak 100 kali dan hasilnya dirata-rata.

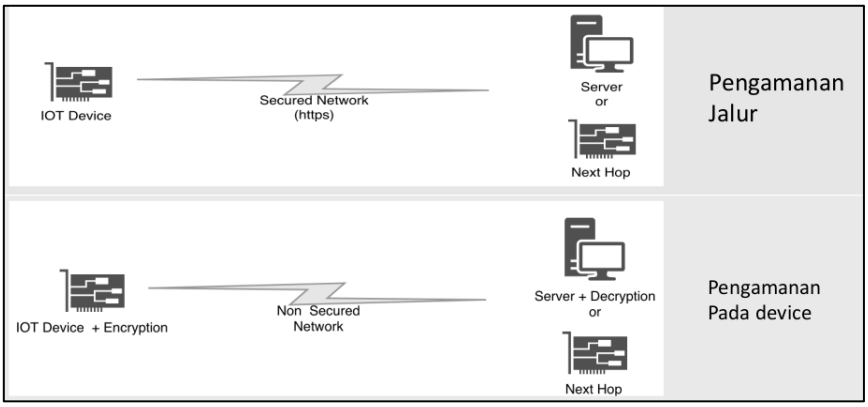
3. Pengujian Kinerja Memori

Pengujian kinerja memori dilakukan untuk mengetahui jumlah memori yang dibutuhkan dalam proses enkripsi data pada prangkat mikrokontroler NodeMCU menggunakan fungsi ESP.getFreeHeap(). Fungsi ESP.getFreeHeap() merupakan fungsi khusus yang disediakan dari *library* ESP32 *board*. Fungsi ini ditempatkan

pada fungsi loop() utama dari kode program (*sketch*) untuk mengetahui jumlah memori yang dibutuhkan dalam proses enkripsi data. Fungsi loop() merupakan fungsi utama dan secara *default* harus ada dalam kode program. Sekenario dilakukan sebanyak 100 kali dan hasilnya dirata-rata.

4. Pengujian Keamanan

Pengujian keamanan dilakukan untuk memastikan sistem telah memenuhi mekanisme keamanan CIA (*Confidentiality, Integrity, Availability*). Pengujian ini berfokus pada unsur *confidentiality* atau kerahasiaan data yang dikirim melalui jaringan *wifi*. Berdasarkan penelitian[2] memperlihatkan skema dalam pengamanan data IoT dapat dilihat pada Gambar 3.18.



Gambar 3. 18 Skema Pengamanan data pada IoT

Pada Gambar 3.18 diperoleh bahwa skema pengamanan data IoT meliputi pengamanan pada sisi jalur komunikasi data menggunakan HTTPS dan pengamanan pada sisi *device* atau perangkat IoT itu sendiri.

Pengujian yang dilakukan pada penelitian ini berfokus pada pengamanan pada sisi *device*. Proses pengujian menggunakan Wireshark dengan melakukan *sniffing* terhadap paket yang lewat melalui *protocol* HTTP, kemudian dilakukan analisis. Hasil pengujian yang diharapkan nantinya adalah dapat mengamankan data yang dikirim, sehingga data yang ditampilkan nantinya dalam bentuk *cipher text* atau data yang tidak dapat dibaca.

Apabila sistem telah berjalan dengan baik dan memenuhi sekenario uji, maka akan dilakukan ke tahap akhir, yakni dokumentasi laporan. Jika sistem tidak berjalan

dengan baik dan tidak memenuhi sekenario uji, maka akan dilakukan perbaikan yang dimulai dari perancangan sistem.

3.6 Dokumentasi Laporan

Pada tahap dokumentasi dan laporan, hasil dari pengujian sistem akan didokumentasikan dan diambil kesimpulan berdasarkan dokumentasi tersebut. Kesimpulan yang telah didapatkan akan dapat digunakan sebagai acuan untuk pengembangan selanjutnya.

3.7 Jadwal Kegiatan

Estimasi waktu yang digunakan dalam proses pengembangan sistem pada penelitian ini yaitu selama kurang lebih 10 minggu. Jadwal kegiatan pengembangan sistem dapat dilihat pada Tabel 3.4.

Tabel 3. 4 Jadwal kegiatan

No	Kegiatan	Waktu (Bulan)										Keterangan
		I	II	III	IV	V	VI	VII	VIII	IX	X	
1	Analisa											Analisa Kebutuhan
2	Perancangan											Perancangan Sistem
3	Implementasi											Implementasi Sistem
4	Pengujian											Pengujian Sistem
5	Dokumentasi											Dokumentasi Sistem

DAFTAR PUSTAKA

[1] "Arduino - Sketch." [Online]. Available: https://www.arduino.cc/en/tutorial/sketch.[Accessed:01-Jun-2020].

[2] A. Muttaqin and Royyannuur, "Mengamankan IoT Device berbasis SoC ESP8266 Menggunakan Lightweight Encryption AES-256", Univ. Brawijaya, 2018.

[3] A. Prameshwari and N. P. Sastra, "Implementasi Algoritme Advanced Encryption Standard (AES) 128 Untuk Enkripsi dan Dekripsi File Dokumen", *Eksplora Inform.*, vol. 8, no. 1, p. 52, 2018.

[4] C. Lung and R. Munir, "Studi Dan Implementasi Advanced Encryption Standard Dengan Empat Mode Operasi Block Cipher," *Dep. Tek. Inform. Inst. Teknol. Bandung*,pp. 1–10, 1997.

[5] FIPS, *Announcing the Advanced Encryption Standard (AES)*, 2001.

[6] IEC 62591, *Wireless communication network and communication profiles – WirelessHART*, 2.0. 2006.

[7] K. Joyoputro, A. Kusyanti, and F. A. Bakhtiar, "Implementasi Algoritme Kriptografi Lizard untuk Mengamankan Pengiriman Data Menggunakan Arsitektur Web Service REST pada Mikrokontroler NodeMCU", vol. 2, no. 12, pp. 6292–6299, 2018.

[8] M. Gupta, M. Abdelsalam, S. Khorsandroo, and S. Mittal, "Security and Privacy in Smart Farming: Challenges and Opportunities," *IEEE Access*, vol. 8, no. February, pp. 34564–34584, 2020.

[9] M. Hakiki, L. Akbar and M. Misbahuddin, "Perbandingan Kinerja Algoritme Kriptografi RSA DAN AES Untuk Keamanan Informasi Perangkat Komunikasi Zigbee", *Univ. Mataram Repos.*, p. 10, 2019.

[10] M. Jamalullail, "Implementasi Kriptografi Pada Komunikasi Perangkat IOT Menggunakan Advanced Encryption Standard (AES)", *Univ. Gadjah Mada*, 2019.

[11] M. P. T. Sulistyanto, D. A. Nugraha, N. Sari, N. Karima, and W. Asrori, "Implementasi IoT (Internet of Things) dalam Pembelajaran di Universitas Kanjuruhan Malang", *SMARTICS J.*, vol. 1, no. 1, pp. 20–23, 2015.

[12] M. Yunus, "Rancang Bangun Sistem Absensi Karyawan Menggunakan Rfid dan Pengenalan Wajah Di Pt. Metro Permata Raya", Univ. Komputer Indonesia, 2019.

[13] N. P. Isnanta, "Analisa Trafik Bandwidth Menggunakan Aplikasi Wireshark Pada Satuan Brimob Polda Jawa Timur," *Inst. Bisnis dan Inform. Stikom Surabaya*, 2016.

[14] P. Pujiono, "Implementasi algoritme aes dan modifikasi vigenere untuk pengamanan pesan sms dengan nomor pengirim dan penerima sebagai kunci tambahan," pp. 1–8, 2015.

[15] Pujianto Yulius Rio, "Perancangan dan Implementasi Aplikasi Kriptografi Algoritme AES-128 Pada File Dokumen," *Univ. Kristen SatyaWacana Salatiga*, 2016.

[16] R. Kristoforus JB, "Implementasi Algoritme Rijndael untuk Enkripsi dan Dekripsi pada Citra Digital", *Semin. Nas. Apl. Teknol. Inf. 2012*, vol. 2012, no. Snati, pp. 15–16, 2012.

[17] R. Munir, *Bahan Kuliah ke-13 Kriptografi, Dep. Tek. Inform. Inst. Teknol. Bandung*, 2004.

[18] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things", *IEEE Computer*, vol. 44, pp. 51 -58, 2011.

[19] R. Sarno, "Analisis dan Desain Berorientasi Servis untuk Aplikasi Manajemen Proyek", *Andy*, 2012.

[20] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures", *Ph.D. disertasi, Univ. Of California*, 2000.

[21] W. Wedashwara, C. Ahmadi, and I. W. A. Arimbawa, "Sequential fuzzy association rule mining algorithm for plants environment classification using internet of things," *in AIP Conference Proceedings*, 2019, vol. 2199, pp. 1–10.

[22] Y. Abidi, “NodeMcu vs Wemos D1 Mini: Which microcontroller should you choose?,” 2019. [Online]. Available: https://candid.technology/nodemcu-vs-wemos-d1-mini-which-microcontroller-comes-out-ahead/. [Accessed: 06-Jun-2020].

LAMPIRAN

KEMENTERIAN RISET,TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS MATARAM
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA
JalanMajapahit No. 62 Mataram 83125 Telpon (0370) 636126 – Fax (0370) 636523

LEMBAR BIMBINGAN TA

NIM : F1D 016 076
Nama Mahasiswa : Rhomy Idris Sardi
Judul : Implementasi Algoritma AES untuk Mengamankan Pengiriman Data Menggunakan Arsitektur *Web Service* REST pada Mikrokontroler NodeMCU
Dosen Pembimbing 1 : Ariyan Zubaidi, S.Kom., M.T.

No	Tanggal	Uraian	Paraf Pembimbing
1.	4 Desember 2019	- Perbaiki latar belakang - perbaiki rumusan masalah	
2.	6 Desember 2019	- perbaiki batasan masalah	
3.	13 Februari 2020	- Lanjut Bab II	
4.	19 Februari 2020	- Perbaiki Tinjauan Pustaka - Perbaiki Daftar Pustaka - Lanjut Bab III	
5.	24 Maret 2020	- perbaiki metodologi penelitian - perbaiki tata tulis	
6.	29 April 2020	- perbaiki penjelasan alur proses pada flowchart - perbaiki tinjauan pustaka	
7.	27 April 2020	- penyajian lebih sebrik - perbaiki tinjauan pustaka	

Mataram, 30 April 2019
Dosen Pembimbing 1,

Ariyan Zubaidi, S.Kom., M.T.
NIP. 19860913 201504 1 001

KEMENTERIAN RISET,TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS MATARAM
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA
JalanMajapahit No. 62 Mataram 83125 Telpon (0370) 636126 – Fax (0370) 636523

LEMBAR BIMBINGAN TA

NIM : F1D 016 076
Nama Mahasiswa : Rhomy Idris Sardi
Judul : Implementasi Algoritma AES untuk Mengamankan Pengiriman Data Menggunakan Arsitektur *Web Service* REST pada Mikrokontroler NodeMCU
Dosen Pembimbing 2 : Andy Hidayat Jatmika, S.T., M.Kom.

No	Tanggal	Uraian	Paraf Pembimbing
1	5-03-2020	BAB I : - Cari data yg mau di enkripsi apa saja. - Perbaiki latar belakang	
2	13-03-2020	BAB I : - Buat alur cerita di latar belakang sehingga fokus penelitian jelas!	
3	20-03-2020	BAB I : OK BAB II : - Teori yg tdk perlu dihapus saja. - Perbaiki tata tulis	

Mataram, 21 April 2019
Dosen Pembimbing 2,

Andy Hidayat Jatmika, S.T., M.Kom.
NIP. 19831209 201212 1 001

KEMENTERIAN RISET,TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS MATARAM
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA
JalanMajapahit No. 62 Mataram 83125 Telpon (0370) 636126 – Fax (0370) 636523

LEMBAR BIMBINGAN TA

NIM : F1D 016 076
Nama Mahasiswa : Rhomy Idris Sardi
Judul : Implementasi Algoritma AES untuk Mengamankan Pengiriman Data Menggunakan Arsitektur *Web Service* REST pada Mikrokontroler NodeMCU
Dosen Pembimbing 2 : Andy Hidayat Jatmika, S.T., M.Kom.

No	Tanggal	Uraian	Paraf Pembimbing
4	3-04-2020	BAB II : - Perbaiki sitasi - Perbaiki tinjauan pustaka	
5	8-04-2020	BAB II : OK BAB III : - Tambahkan tahapan penelitian	
6	19-04-2020	BAB III : - Perbaiki flowchart tahapan penelitian dan tata tulis	
7	21-04-2020	BAB III : - Lanjut tahapan pengujian BAB III : OK	

Mataram, 21 April 2019
Dosen Pembimbing 2,

Andy Hidayat Jatmika, S.T., M.Kom.
NIP. 19831209 201212 1 001