PENGAMANAN INTERNET OF THINGS MENGGUNAKAN ALGORITME SIMON DAN SPECK UNTUK LAYANAN KERAHASIAAN DATA

WAWAN SETYADI

DEPARTEMEN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT PERTANIAN BOGOR
BOGOR
2019

PERNYATAAN MENGENAI SKRIPSI DAN SUMBER INFORMASI SERTA PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa skripsi berjudul Pengamanan *Internet of Things* Menggunakan Algoritme SIMON dan SPECK untuk Layanan Kerahasiaan Data adalah benar karya saya dengan arahan dari komisi pembimbing dan belum diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka di bagian akhir skripsi ini.

Dengan ini saya melimpahkan hak cipta dari karya tulis saya kepada Institut Pertanian Bogor.

Bogor, April 2019

Wawan Setyadi
NIM G64140088

ABSTRAK

WAWAN SETYADI. Pengamanan *Internet of Things* Menggunakan Algoritme SIMON dan SPECK untuk Layanan Kerahasiaan Data. Dibimbing oleh SHELVIE NIDYA NEYMAN.

Internet of Things (IoT) memiliki potensi yang besar dan semakin berperan penting dalam kehidupan manusia. Potensi besar IoT juga membawa tantangan yang besar, salah satunya adalah keamanan informasi. Ketika informasi pada jaringan IoT mengalami masalah, seluruh objek yang terhubung dalam jaringan akan ikut terpengaruh dan mengganggu jalannya sistem. Penelitian ini mencoba menjawab tantangan tersebut dengan menerapkan aloritme SIMON dan SPECK, untuk membuat sebuah layanan keamanan kerahasiaan data pada lingkungan simulasi sistem IoT sederhana. Sistem IoT dibuat menggunakan *microcontroller* Arduino Uno yang umum digunakan di dunia nyata. Analisa penelitian ini dilakukan dengan mengukur kinerja algoritme pada *microcontroller*, baik berdasarkan waktu eksekusi, maksimal panjang data yang dapat diproses, besar memori yang digunakan, maupun ketahanannya terhadap serangan. Hasil penelitian ini menunjukkan bahwa algoritme SPECK memiliki kinerja yang lebih baik pada perangkat Arduino Uno dibandingkan algoritme SIMON, dan kedua algoritme tersebut dapat diterapkan pada sistem IoT bersumber daya rendah untuk pembuatan layanan keamanan kerahasiaan data.

Kata kunci: *Internet of Things*, keamanan informasi, kerahasiaan data, SIMON, SPECK

ABSTRACT

WAWAN SETYADI. Secure Internet of Things (IoT) Using SIMON and SPECK Algorithm for Data Secuity Services. Supervised by SHELVIE NIDYA NEYMAN.

The Internet of Things (IoT) has great potential and increasingly plays an important role in human life. Great potential of IoT also brings great challenges, one of which is information security. When information on the IoT network experiences problems, all objects connected in the network will be affected and disrupt the running of the system. This study tries to answer this challenge by applying the SIMON and SPECK algorithm, to create a data confidentiality security service in a simple IoT system simulation environment. The IoT system is made using the Arduino Uno microcontroller that is commonly used in the real world. Analysis of this study is done by measuring the performance of algorithms on microcontrollers, both based on execution time, the maximum length of data that can be processed, the amount of memory used, and its resistance to attacks. The results of this study indicate that the SPECK algorithm has better performance on Arduino Uno devices than SIMON algorithms, and both of these algorithms can be applied to low-resource IoT systems for the creation of data security confidentiality services.

Keywords: data confidentiality, information security, Internet of Things, SIMON, SPECK

PENGAMANAN INTERNET OF THINGS MENGGUNAKAN ALGORITME SIMON DAN SPECK UNTUK LAYANAN KERAHASIAAN DATA

Skripsi
sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer
pada
Departemen Ilmu Komputer

DEPARTEMEN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT PERTANIAN BOGOR
BOGOR
2019

Penguji:
1 Dr Eng. Heru Sukoco, SSi MT
2 Auriza Rahmad Akbar, SKomp MKom

Judul Skripsi: Pengamanan *Internet of Things* Menggunakan Algoritme SIMON dan SPECK untuk Layanan Kerahasiaan Data

Nama : Wawan Setyadi

NIM : G64140088

Disetujui oleh

Dr Shelvie Nidya Neyman, SKom MSi
Pembimbing

Diketahui oleh

Prof Dr Ir Agus Buono, MSi MKom
Ketua Departemen

Tanggal Lulus: 09 APR 2019

PRAKATA

Puji dan syukur penulis panjatkan kepada Allah *subhanahu wa ta'ala* atas segala karunia-Nya sehingga karya ilmiah ini berhasil diselesaikan. Tema yang dipilih dalam penelitian yang dilaksanakan sejak bulan Januari 2018 ini ialah keamanan *Internet of Things*, dengan judul Pengamanan *Internet of Things* Menggunakan Algoritme SIMON dan SPECK untuk Layanan Kerahasiaan Data.

Tak lupa juga penulis sampaikan ucapan terima kasih kepada:

1 Ayah, ibu, serta seluruh keluarga atas doa dan dukungannya kepada saya.
2 Ibu Dr Shelvie Nidya Neyman, SKom MSi selaku pembimbing yang memberikan saran, bimbingan, dan arahan selama penelitian berlangsung.
3 Bapak Dr Eng Heru Sukoco, SSi MT dan Auriza Rahmad Akbar, SKomp MKom sebagai penguji.
4 Teman-teman mahasiswa Ilmu Komputer FMIPA IPB angkatan 51 yang turut memberikan dukungan kepada penulis dalam menyelesaikan penelitian.

Semoga karya ilmiah ini bermanfaat.

Bogor, April 2019

Wawan Setyadi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

PENDAHULUAN

Latar Belakang

Kemunculan *Internet of Things* (IoT) dipercaya sebagai generasi berikutnya dari Internet dan dapat mengubah segalanya. Saat ini ada lebih dari 12 miliar alat pintar yang dapat terhubung ke Internet, dan diperkirakan pada tahun 2020 nanti alat pintar yang terhubung akan berjumlah 6 sampai 7 kali lebih banyak dibandingkan jumlah manusia (Keerthana dan Parveen 2017). IoT sendiri terdiri atas perangkat digital, manusia, layanan dan objek fisik lainnya yang dapat saling terhubung, berinteraksi, serta bertukar informasi tentang dirinya dan lingkungannya melalui jaringan internet. Hal ini dapat membuat hidup manusia lebih mudah melalui lingkungan yang lebih adaptif dan responsif terhadap kebutuhan manusia (Babar *et al.* 2011). Semakin lama, IoT semakin berperan penting dalam kehidupan manusia. Dalam beberapa tahun terakhir, IoT telah dimanfaatkan untuk mengembangkan beberapa aplikasi dengan infrastruktur yang berbeda seperti untuk logistik, manufaktur, kesehatan, pengawasan industri, bahkan peralatan rumah tangga yang ada disekitar kita (Li dan Xu 2017).

Besarnya potensi yang dimiliki oleh IoT memberikan banyak tantangan yang harus dilalui oleh para pengembangnya, seperti *massive scaling*, arsitektur, keandalan, keterbukaan, privasi, keamanan dan lain-lain (Stankovic 2014). Di antara beberapa tantangan tersebut, yang menjadi fokus utama saat ini adalah tantangan keamanan. Dengan banyaknya objek yang terhubung dalam jaringan IoT, informasi yang dipertukarkan menjadi semakin banyak dan heterogen. Ketika informasi tersebut dicuri atau terganggu saat proses pengiriman, seluruh objek yang terhubung dengan jaringan IoT tersebut akan ikut terpengaruh dan mengganggu jalannya sistem (Jing *et al.* 2014). Maka dari itu, kerahasiaan data perlu dijaga agar pihak yang tidak berwenang tidak dapat melihat atau mengubah data pada suatu sistem.

Permasalahan utama yang terdapat dalam penerapan sistem keamanan pada sistem berbasis IoT adalah terbatasnya sumber daya perangkat IoT itu sendiri. Perangkat IoT biasanya memiliki kemampuan komputasi lambat, kapasitas penyimpanan yang kecil, *bandwidth* yang rendah dan memiliki protokol *wireless* berbeda untuk setiap jenis perangkat. Sedangkan algoritme kriptografi kontemporer membutuhkan kemampuan komputasi cepat, kapasitas penyimpanan yang besar, *bandwidth* yang tinggi dan dirancang untuk satu jenis protokol saja (Hossain *et al.* 2017).

Penelitian sebelumnya yang dilakukan oleh Dinu *et al*. (2015) mengenai kerangka pembandingan sandi blok pada berbagai *embedded platform* menunjukkan bahwa, algoritme kriptografi yang umum digunakan seperti AES mengonsumsi sumber daya yang besar dan kurang sesuai diterapkan pada perangkat bersumber daya rendah. Setelah dibandingkan dengan beberapa algoritme lain, SIMON dan SPECK merupakan algoritme kriptografi ringan yang terkecil dan tercepat pada semua *platform*. Dari penelitian tersebut, penelitian ini mencoba menerapkan dan membandingkan algoritme SIMON dan SPECK, untuk mengamankan sistem berbasis IoT dengan layanan kerahasiaan data sebagai lingkungan simulasi.

Perumusan Masalah

Masalah yang diangkat dalam penelitian ini adalah perlunya perlindungan untuk menjaga kerahasiaan data pada sistem IoT, dengan sumber daya yang terbatas. Oleh karena itu diperlukan suatu layanan keamanan kerahasiaan data yang kuat dan ringan, sehingga layanan dapat diterapkan pada semua sistem berbasis IoT. Algoritme kriptografi yang dapat digunakan untuk memberikan layanan kerahasiaan data tersebut adalah kriptografi kunci simetri.

Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1 Mengembangkan prototipe sistem berbasis IoT sederhana yang memiliki layanan kerahasiaan data sebagai lingkungan simulasi.
2 Mengukur dan membandingkan kinerja algoritme kriptografi kunci simetri SIMON dan SPECK, sebagai perangkat pemberi layanan kerahasian data pada *microcontroller* yang digunakan.

Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan pilihan yang lebih luas dalam pembuatan layanan keamanan kerahasiaan data pada *microcontroller* atau perangkat IoT bersumber daya rendah lainnya.

Ruang Lingkup Penelitian

Ruang lingkup yang digunakan dalam penelitian ini adalah:

1 Data yang dikirim oleh perangkat IoT ke aplikasi merupakan alamat IP dan data GPS berupa *latitude* dan *longitude* dari lokasi *microcontroller* saat ini.
2 Perangkat IoT yang digunakan dibuat menggunakan *microcontroller* bersumber daya rendah dan sering digunakan dalam lingkungan kerja yaitu Arduino Uno.
3 Layanan pada lingkungan simulasi sistem tidak mencakup distribusi kunci.

TINJAUAN PUSTAKA

Keamanan Informasi

Keamanan Informasi adalah upaya perlindungan dari berbagai macam ancaman untuk memastikan keberlanjutan bisnis, meminimalisir resiko bisnis, dan meningkatkan investasi dan peluang bisnis (Menezes *et al*. 1996). Adapun tujuan dari keamanan informasi adalah sebagai berikut:

1 Kerahasiaan adalah suatu layanan yang digunakan untuk menjaga isi informasi dari semua yang tidak berwenang memilikinya.

2 Integritas data adalah suatu layanan yang digunakan untuk menjaga agar isi informasi tidak diubah oleh pihak yang tidak berwenang.
3 Autentikasi adalah suatu layanan yang yang digunakan untuk mengidentifikasi entitas dan informasi itu sendiri.
4 Non-repudiasi adalah suatu layanan yang ditujukan untuk mencegah terjadinya pelanggaran kesepakatan yang telah dibuat sebelumnya oleh entitas.

Algoritme SIMON dan SPECK

SIMON dan SPECK merupakan algoritme kriptografi kunci simetri yang diajukan ke publik pada Juni 2013 oleh sekelompok peneliti di NSA Amerika serikat. Tujuan dibuatnya algoritme ini adalah untuk membuat sebuah algoritme yang cukup fleksibel untuk berjalan dengan baik pada perangkat yang terbatas (Beaulieu *et al.* 2015).

Mengutip dari Beaulieu *et al.* (2015), agar algoritme kriptografi dapat berguna semaksimal mungkin algoritme tersebut harus cukup fleksibel untuk diterapkan secara efisien pada berbagai perangkat, dan juga memungkinkan untuk berbagai implementasi pada satu perangkat. Pada perangkat keras, hal tersebut artinya algoritme dapat mengambil keuntungan dari spesifikasi perangkat yang tersedia baik dalam lingkungan bersumber daya rendah maupun linkungan bersumber daya tinggi. Pada perangkat lunak, hal tersebut artinya algoritme mampu menggunakan sumber daya seminimal mungkin dan mengeluarkan hasil sebaik mungkin. Untuk memberikan fleksibilitas tersebut algoritme SIMON dan SPECK mendukung beberapa parameter pasangan ukuran blok dan ukuran kunci yang dapat digunakan seperti yang terlihat pada Tabel 1.

Tabel 1 Pasangan parameter algoritme SIMON dan SPECK

Ukuran blok (bit)	Ukuran kunci (bit)	Ukuran *word* (bit)	Jumlah *key words* (m)	Urutan konstanta SIMON (z_j)	Jumlah ronde SIMON	Jumlah ronde SPECK
32	64	16	4	0	32	22
48	72	24	3	0	36	22
	96		4	1	36	23
64	96	32	3	2	42	26
	128		4	3	44	27
96	96	48	2	2	52	28
	144		3	3	54	29
128	128	64	2	2	68	32
	192		3	3	69	33
	256		4	4	72	34

Sumber: Beaulieu *et al.* (2015)

Sebelum melakukan proses enkripsi dan dekripsi, algoritme SIMON dan SPECK memerlukan penjadwalan kunci untuk menentukan kunci apa yang akan

digunakan setiap rondenya. Penjadwalan kunci algoritme SIMON menggunakan konstanta ronde untuk mengeliminasi *slide properties* dan *circular shift symmetries* yang terjadi saat proses enkripsi maupun dekripsi. Cara penjadwalan kunci pada algoritme SIMON tergantung pada pasangan ukuran blok dan kunci, seperti yang terlihat pada persamaan 1 dan 2.

$$k_0..k_{m-1} = key[m-1]..key[0] \quad (1)$$

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})\, S^{-3}\, k_{i+1}, & m = 2 \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})\, S^{-3}\, k_{i+2}, & m = 3 \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})\, (S^{-3}\, k_{i+2} \oplus k_{i+1}), & m = 4 \end{cases} \quad (2)$$

Sedangkan untuk algoritme SPECK penjadwalan kunci dilakukan menggunakan fungsi ronde untuk menghasilkan kunci ronde k_i, seperti yang terlihat pada persamaan 3, 4, 5 dan 6.

$$k_0 = key[m-1] \quad (3)$$

$$l_0..l_{m-2} = key[m-2]..key[0] \quad (4)$$

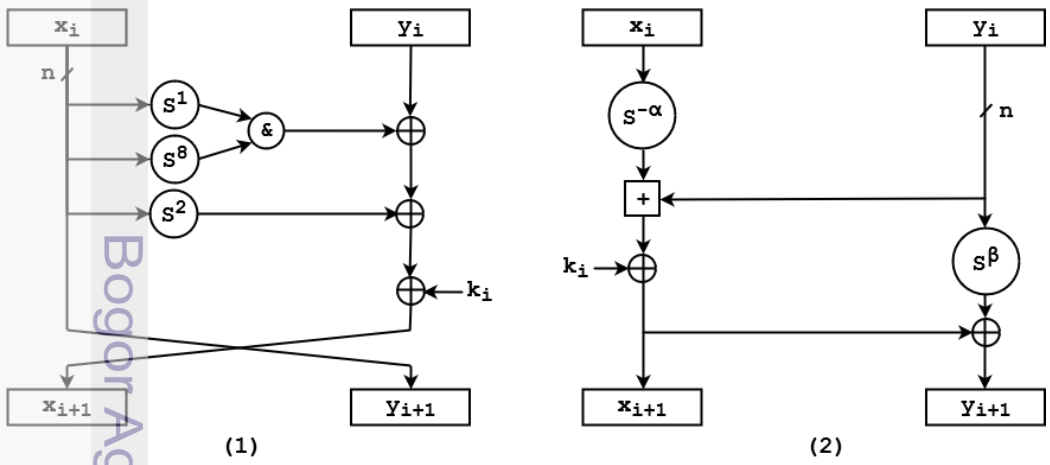
$$l_{i+m-1} = (k_i + S^{-\alpha} l_i) \oplus i \quad (5)$$

$$k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1} \quad (6)$$

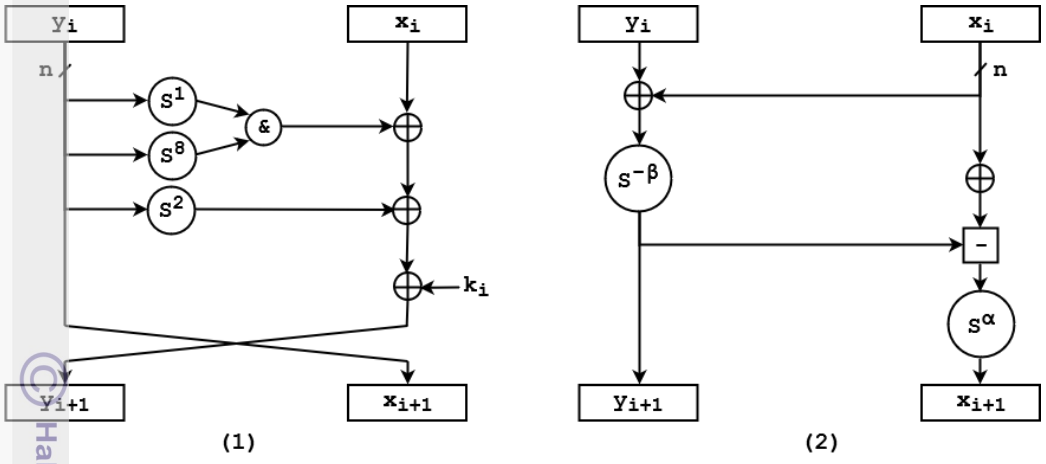
dengan

$\oplus$ = operasi *bitwise* XOR
S^j = geser kiri melingkar sebanyak j bits
$(z_j)_i$ = konstanta SIMON ronde ke-i index ke-j
α,β = jumlah rotasi pada algoritme SPECK (α=7, β=2 jika ukuran blok 32 bit dan α=8, β=3 jika ukuran blok lebih dari 32 bit)
$key[i]$ = kunci blok ke-i
l_i = variabel sementara ke-i
k_i = kunci ronde ke-i
c = *bit mask*

Setelah dilakukan penjadwalan kunci, tahap beriutnya adalah melakukan fungsi ronde sebanyak jumlah rondenya. Fungsi ronde enkripsi algoritme SIMON dan SPECK digambarkan pada Gambar 1. Sedangkan fungsi ronde dekripsi digambarkan pada Gambar 2.



Gambar 1 Fungsi ronde enkripsi untuk algoritme SIMON (1) dan SPECK (2)



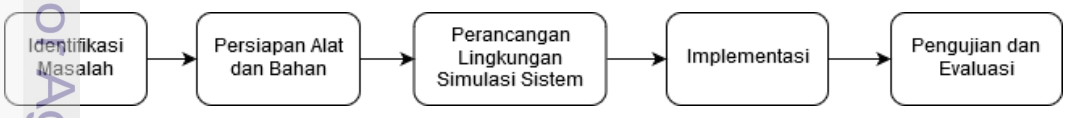
Gambar 2 Fungsi ronde dekripsi untuk algoritme SIMON (1) dan SPECK (2)

Pada umumnya kriptografi sandi blok seperti AES dan DES dilakukan menggunakan *substitution box* (Sbox) agar dapat dibentuk menjadi *substitution-permutation networks* (SPNs). Keunggulan dari SPNs adalah dia memungkinkan argumen keamanan yang relatif sederhana dan cukup kuat secara kriptografis, tetapi cukup berat pada perangkat yang terbatas. Berdasarkan pertimbangan tersebut, algoritme SIMON dan SPECK dibuat tanpa menggunakan Sbox tetapi menggunakan fungsi ronde berbasis permutasi Feistel yang memberikan keseimbangan antara *linear diffusion* dan *nonlinear confusion operations* (Beaulieu *et al.* 2015).

Algoritme SIMON dirancang menggunakan operasi XOR, AND, dan *left circular shift* pada fungsi rondenya, sehingga sangat mendukung implementasi pada perangkat keras. Sedangkan algoritme SPECK dirancang menggunakan operasi XOR, penjumlahan modular, serta *left and right circular shift* pada fungsi rondenya, sehingga sangat mendukung implementasi pada perangkat lunak. Contoh hasil *tracing* dari algoritme SIMON dan SPECK dapat dilihat pada Lampiran 1 dan Lampiran 2.

METODE

Penelitian ini akan menerapkan dan membandingkan kinerja algoritme SIMON dan SPECK untuk mengamankan kerahasiaan data pada perangkat IoT. Penelitian dilakukan secara bertahap Gambar 3. Penelitian diawali dengan identifikasi masalah, persiapan alat dan bahan, perancangan lingkungan simulasi sistem, implementasi, serta pengujian dan evaluasi hasil. Hasil akhir dari penelitian ini adalah prototipe sistem berbasis IoT sederhana siap pakai yang memiliki layanan kerahasiaan data.



Gambar 3 Tahapan penelitian

Identifikasi Masalah

Pada tahap ini diidentifikasi permasalahan yang ada pada sistem berbasis IoT saat ini dan cara mengatasinya. Tahap ini diperlukan agar sistem yang dihasilkan dapat berguna dan benar-benar menyelesaikan masalah yang ada di lapangan. Kegiatan-kegiatan yang dilakukan dalam tahap ini adalah mengumpulkan dan memperdalam materi yang berhubungan dengan penelitian, merumuskan masalah, menentukan tujuan, menentukan manfat, dan membatasi ruang lingkup yang akan digunakan dalam penelitian.

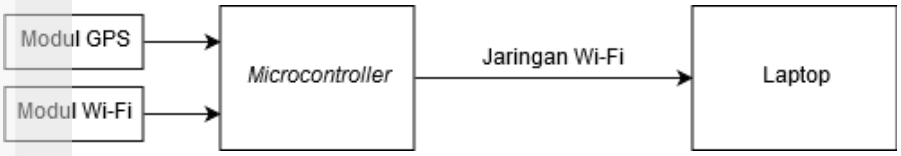
Persiapan Alat dan Bahan

Pada tahap ini peneliti merancang perangkat keras dan perangkat lunak yang dibutuhkan dalam penelitian, berdasarkan identifikasi masalah yang telah dilakukan sebelumnya. Dalam pembuatan lingkungan simulasi sistem, dibutuhkan alat dan bahan berikut:

- Satu buah *microcontroller* untuk melakukan komputasi pada perangkat IoT,
- Satu buah modul GPS untuk mendeteksi lokasi perangkat IoT saat ini,
- Satu buah modul Wi-Fi untuk menghubungkan perangkat IoT dengan aplikasi melalui jaringan Wi-Fi,
- Sebelas buah kabel *jumper* untuk menghubungkan *microcontroller* dengan modul-modul yang digunakan,
- Satu buah breadboard untuk menyusun rangkaian modul dan kabel,
- Satu buah laptop untuk memrogram Arduino Uno dan menjalankan aplikasi deteksi lokasi.

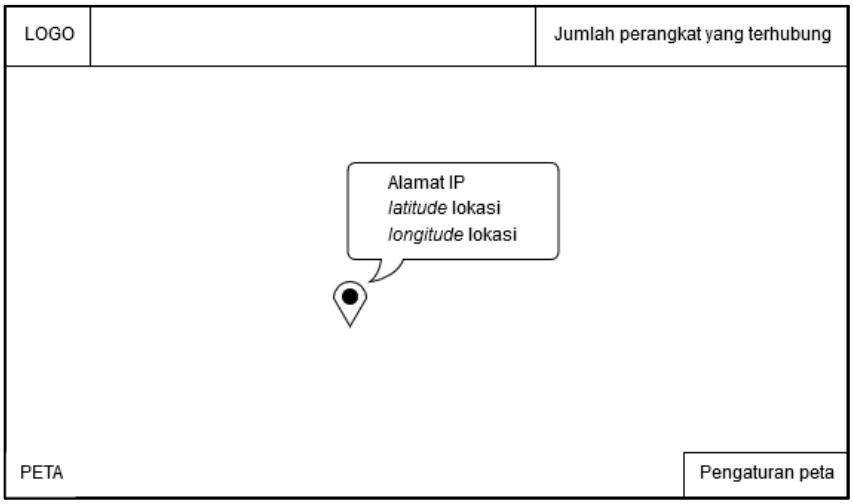
Perancangan Lingkungan Simulasi Sistem

Pada tahap ini dirancang suatu lingkungan simulasi sistem menggunakan alat dan bahan yang telah ditentukan. Rangkaian lingkungan simulasi dibuat sederhana agar lebih mudah dirakit dan dianalisis. Secara umum rangkaian lingkungan simulasi yang dibuat dapat terlihat pada Gambar 4.



Gambar 4 Blok diagram lingkungan simulasi sistem

Modul GPS akan mengambil data lokasi perangkat saat ini berupa *latitude* dan *longitude*, yang kemudian diproses dan dienkripsi oleh *microcontroller*. Data tersebut kemudian dikirim melalui jaringan Wi-Fi ke aplikasi yang ada pada laptop, pada proses pengiriman inilah kerahasiaan data harus terjaga. Data yang diterima kemudian didekripsi dan ditampilkan pada aplikasi deteksi lokasi berbasis sistem informasi geografis agar lebih mudah diamati. Tampilan pada aplikasi deteksi lokasi tersebut secara sederhana dapat terlihat pada Gambar 5.



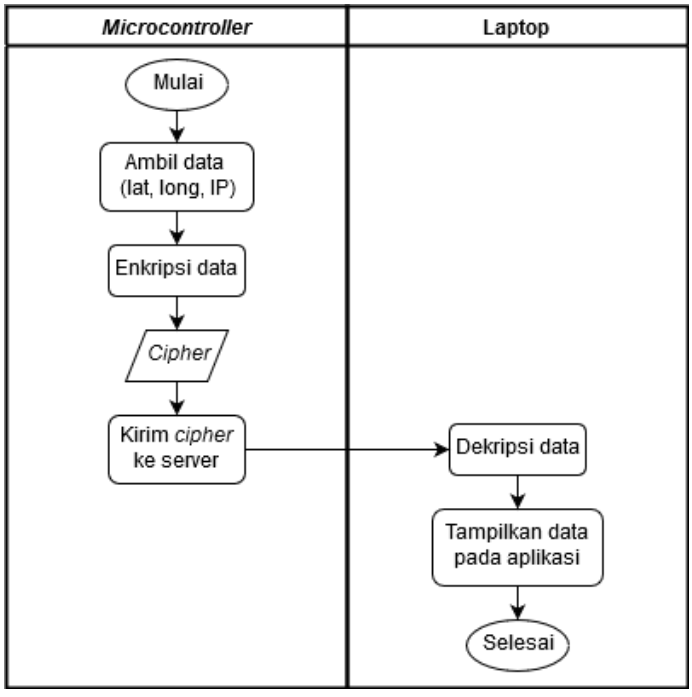
Gambar 5 *Mockup* tampilan pada aplikasi deteksi lokasi

Implementasi

Pada tahap ini diterapkan seluruh bagian sistem keamanan yang telah dirancang sebelumnya ke dalam lingkungan simulasi. Secara garis besar tahap penerapan ini terbagi menjadi tiga bagian, yaitu implementasi perangkat, implementasi kode, dan implementasi aplikasi.

a Implementasi Perangkat

Setiap perangkat dihubungkan satu sama lain sehingga perangkat-perangkat tersebut dapat saling berinteraksi. Secara umum alur kerja sistem dapat dilihat pada Gambar 6.

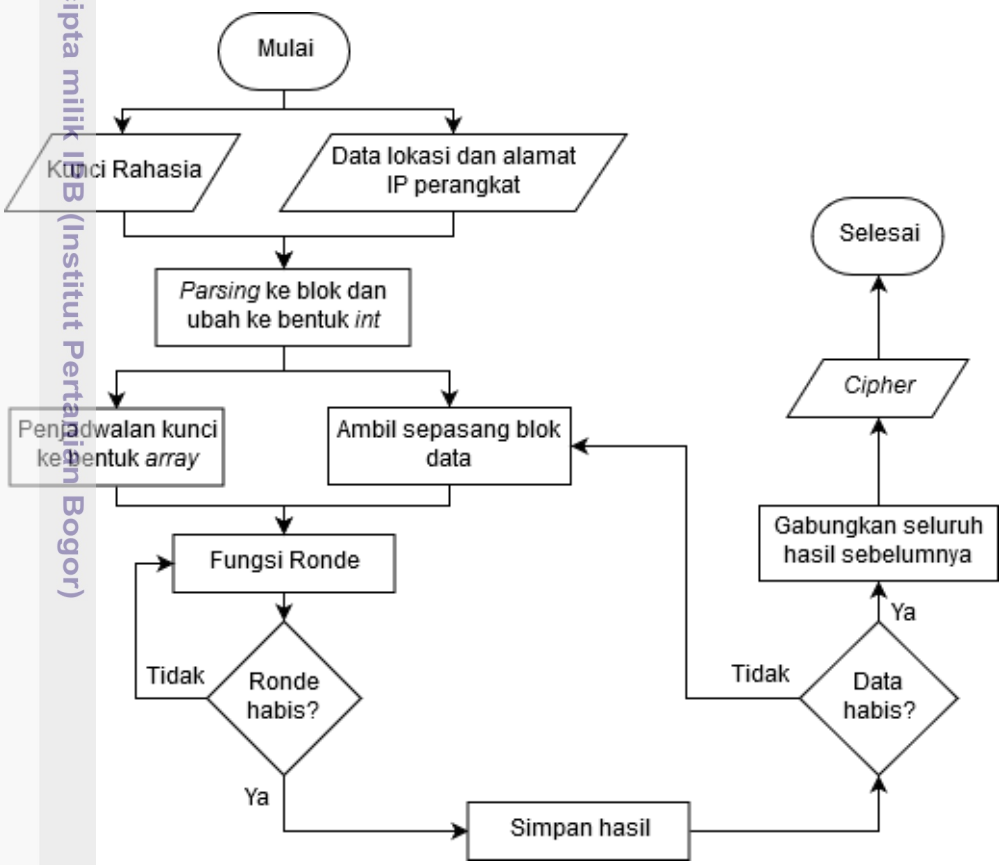


Gambar 6 *Flowchart* sistem secara umum

Microcontroller mengambil data berupa *latitude*, *longitude*, dan alamat IP perangkat yang kemudian dienkripsi menjadi *cipher*. *Cipher* tersebut dikirim ke laptop melalui jaringan Wi-Fi. Setelah menerima *cipher*, laptop akan menderipsinya menjadi data kembali dan menampilkannya pada aplikasi berbasis sistem informas geografis.

b Implementasi Kode

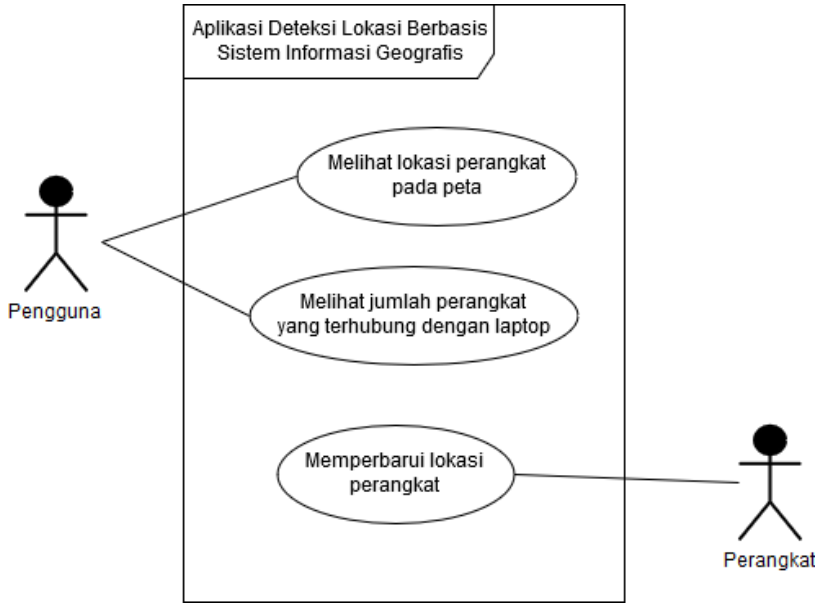
Dibuat program pada laptop pribadi yang nantinya akan diunggah ke *microcontroller* menggunakan IDE. Algoritme program yang dibuat dapat terlihat pada Gambar 7. Umumnya algoritme Simon dan Speck memiliki alur yang sama, dimulai dari mengambil data, menggabungkan data, *parsing* data ke blok, penjadwalan kunci, fungsi ronde, hingga menggabungkan hasil menjadi *cipher*. Hanya saja cara penjadwalan kunci dan fungsi rondenya berbeda tergantung algoritmenya.



Gambar 7 *Flowchart* algoritme SIMON dan SPECK

c Implementasi Aplikasi

Dibuat sebuah aplikasi deteksi lokasi sederhana berbasis sistem informasi geografis sesuai dengan *mockup* yang telah dirancang sebelumnya. Aplikasi tersebut akan terhubung melalui jaringan Wi-Fi ke *microcontroller* dan akan terbarui secara otomatis dalam jangka waktu tertentu. Untuk lebih jelasnya, *use case* aplikasi tersebut dapat dilihat pada Gambar 8.



Gambar 8 *Use case* aplikasi deteksi lokasi

Pengujian dan Evaluasi

Terdapat tiga jenis pengujian yang akan dilakukan terhadap lingkungan simulasi sistem, yaitu pengujian fungsional, pengujian kinerja, dan pengujian keamanan. Pengujian fungsional dilakukan untuk melihat apakah sistem dapat berjalan dengan seharusnya. Pengujian kinerja dilakukan untuk mengukur seberapa baik kinerja masing-masing algoritme pada sistem tersebut. Kinerja dinilai berdasarkan kelajuan kecepatan komputasi terhadap panjang data, maksimal panjang data yang dapat diproses, memori yang digunakan, serta kompleksitas algoritme. Pengujian keamanan dilakukan untuk melihat ketahanan algoritme terhadap serangan.

Lingkungan Pengembangan

Perangkat keras yang digunakan untuk penelitian ini adalah:

1 Arduino Uno sebagai *microcontroller* dengan spesifikasi sebagai berikut:
- *Clock speed* 16MHz
- *Flash memory* 32 KB (0.5 untuk *bootloader*)
- SRAM 2 KB
- EEPROM 1 KB

2 Ublox NEO-6M sebagai modul GPS dengan spesifikasi sebagai berikut:
- Sensitivitas *Hotstart* -156dBm
- Protokol GPS NMEA
- Tingkat pembaruan posisi 5Hz

3 ESP8266-01 sebagai modul Wi-Fi.
- Daya keluaran +19.5dBm
- *Flash memory* 1MB
- Wi-Fi 2.4GHz

4 Laptop pribadi dengan spesifikasi sebagai berikut:
- *Processor* AMD A10-7400P Radeon R6 2.5GHz *Quad Core*
- RAM 4 GB
- HDD 1 TB

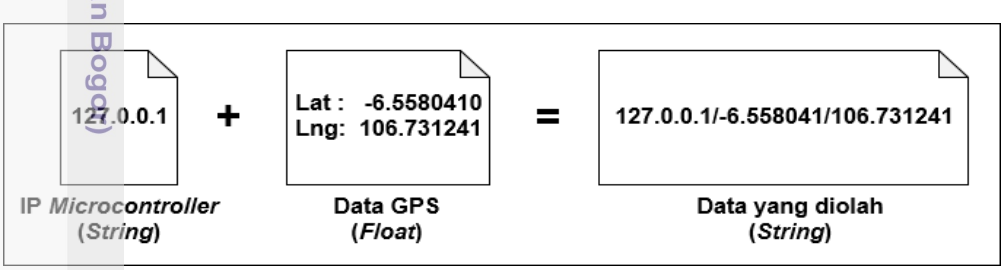
Perangkat lunak yang digunakan untuk penelitian ini adalah:

1 Fritzing 0.9.2 sebagai aplikasi untuk merancang rangkaian perangkat pada lingkungan simulasi.
2 Python 3.5.2 sebagai bahasa pemrograman untuk membuat aplikasi deteksi lokasi pada laptop.
3 MySQL 5.0.12 sebagai *database management system* (DBMS).
4 Google Maps API sebagai *library* untuk pembuatan tampilan sistem informasi geografis pada aplikasi deteksi lokasi.
5 Arduino IDE sebagai aplikasi untuk membuat kode program dan kemudian mengunggahnya ke *microcontroller* Arduino.
6 Wireshark sebagai aplikasi untuk menguji keamanan sistem melalui simulasi *man-in-the-middle attack.*

HASIL DAN PEMBAHASAN

Data Penelitian

Data yang digunakan pada penelitian ini berupa informasi alamat IP *microcontroller* beserta data lokasi *longitude* dan *latitude* yang diperoleh dari modul GPS. Contoh data penelitian dapat dilihat pada Gambar 9.

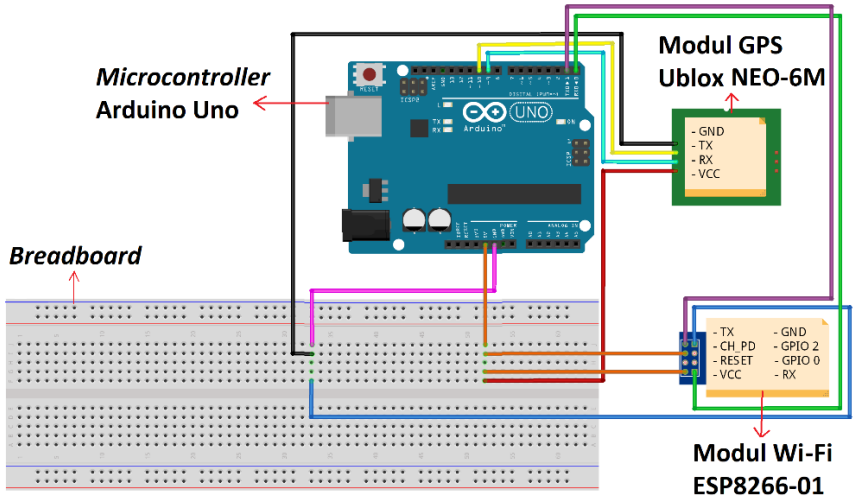


Gambar 9 Contoh data penelitian

Implementasi

1 Implementasi Perangkat

Setiap modul seperti GPS Ublox NEO-6M dan ESP8266-01 dapat terhubung ke Arduino Uno menggunakan kabel jumper. Kabel tersebut dihubungkan pada pin tertentu berdasarkan skema rangkaian seperti yang terlihat pada Gambar 10. Jika kedua modul harus dihubungkan pada pin Arduino Uno yang sama, digunakan *breadboard* untuk membantu membagi dan menghubungkannya.



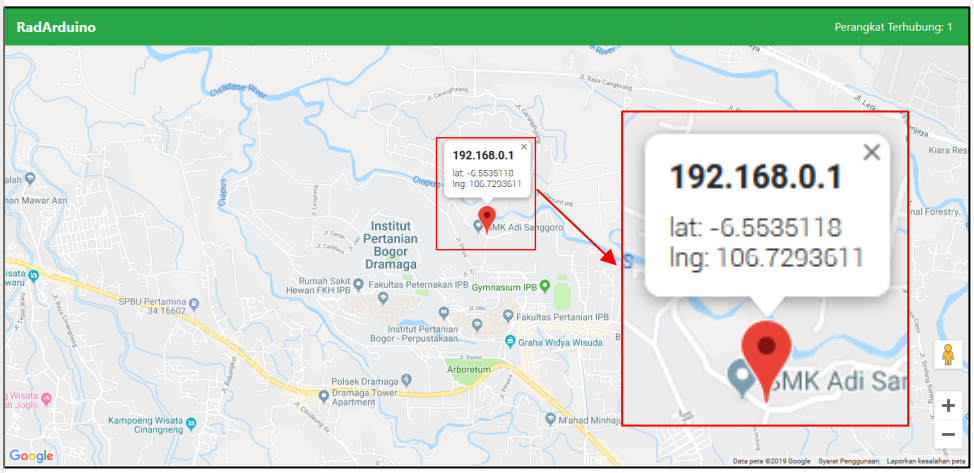
Gambar 10 Skema rangkaian perangkat IoT

2 Implementasi Kode

Setelah setiap modul telah terhubung ke *microcontroller*, dibuat kode program enkripsi untuk masing-masing algoritme sesuai dengan *flowchart* algoritme yang telah dibuat sebelumnya menggunakan bahasa pemrograman *default* untuk Arduino IDE yaitu bahasa C. Kode program selengkapnya dapat dilihat pada Lampiran 3 dan 4 untuk algoritme SIMON serta Lampiran 5 dan 6 untuk algoritme SPECK. Kode program yang telah dibuat tersebut kemudian diunggah ke perangkat yang telah dirangkai sebelumnya.

3 Implementasi Aplikasi

Aplikasi deteksi lokasi dibuat menggunakan bahasa pemrograman Python dan Google Maps API untuk tampilannya serta MySQL sebagai tempat penyimpanan sementara untuk data lokasi perangkat. Hasil tampilan yang telah dibuat dapat dilihat pada Gambar 11. Pin merah di tengah peta merupakan lokasi perangkat saat ini, yang jika ditekan akan menampilkan informasi rinci berupa *latitude*, *longitude*, dan alamat IP perangkat. Lokasi pin akan diperbarui secara otomatis setiap 5 detik sekali.



Gambar 11 Tampilan aplikasi deteksi lokasi

Pengujian dan Evaluasi Hasil

1 Pengujian Fungsional

Setiap perangkat yang digunakan telah diuji sesuai dengan fungsinya dan mendapatkan hasil yang cukup baik. Masing-masing perangkat dapat berfungsi sesuai dengan yang diharapkan, kecuali untuk modul GPS Ublox NEO-6M yang tidak berjalan dengan baik di dalam ruangan. Hal ini disebabkan karena GPS memiliki frekuensi yang besar sehingga buruk untuk perambatan. Selain itu, rasio sinyal terhadap *noise* GPS biasanya dekat dengan ambang batas untuk deteksi, sehingga sinyal tidak kuat ketika masuk ke dalam ruangan. Untuk lebih jelasnya, hasil pengujian fungsional perangkat dapat dilihat pada Tabel 2.

Tabel 2 Hasil pengujian fungsional perangkat keras

Perangkat keras	Fungsi	Keterangan
Arduino Uno	Memproses data dan mengenkripsinya menjadi *cipher*	Data berhasil diproses dan terenkripsi dengan baik
GPS Ublox NEO-6M	Mendapatkan lokasi perangkat saat ini dalam bentuk *latitude* dan *longitude*	Lokasi berhasil didapatkan ketika perangkat berada di luar ruangan
ESP8266-01	Menghubungkan perangkat ke aplikasi melalui jaringan Wi-Fi	*Cipher* berhasil terkirim ke aplikasi melalui jaringan Wi-Fi
Laptop	Mendekripsi *cipher* dan menampilkan data pada aplikasi deteksi lokasi	Data berhasil terdekripsi dan terlihat dengan baik pada aplikasi deteksi lokasi

Setelah perangkat keras selesai diuji, aplikasi deteksi lokasi yang terdapat pada laptop juga diuji berdasarkan *use case* yang telah dibuat sebelumnya. Hasil pengujian dapat dilihat pada Tabel 3. Terlihat bahwa setiap *use case* pada aplikasi dapat berfungsi sesuai dengan yang diharapkan.

Tabel 3 Hasil pengujian *use case* aplikasi deteksi lokasi

Use case	Aktor	Keterangan
Melihat lokasi perangkat pada peta	Pengguna	Lokasi perangkat dapat terlihat dengan jelas dalam betuk pim merah
Melihat jumlah perangkat yang terhubung ke aplikasi deteksi lokasi	Pengguna	Jumlah perangkat yang terhubung ke aplikasi dapat terlihat dengan baik
Memperbarui lokasi perangkat	*Microcontroller*	Lokasi berhasil diperbarui setiap jangka waktu lima detik

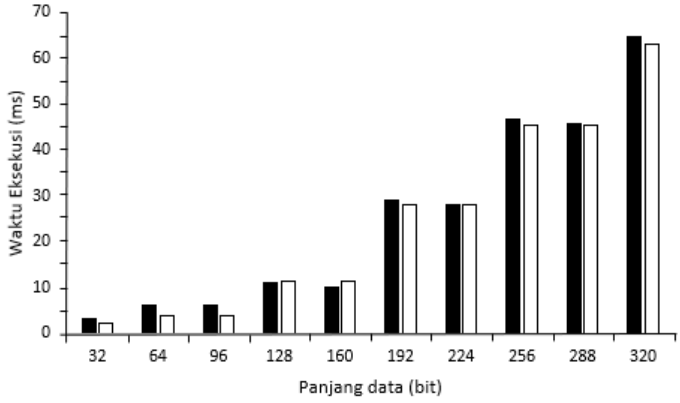
2 Pengujian Kinerja

Pengujian ini dilakukan dalam beberapa tahap, tahap pertama dilakukan untuk mengukur maksimal panjang data yang dapat diproses dan waktu eksekusi yang dibutuhkan oleh masing-masing algoritme. Tahap ini dilakukan dengan menjalankan program enkripsi dan dekripsi menggunakan parameter 64 bit ukuran blok dan 128 bit ukuran kunci. Panjang data akan terus bertambah hingga mencapai maksimal dengan kelipatan 32 bit. Jika panjang data yang diolah tidak memenuhi satu blok, maka akan ditambahkan *padding* hingga blok tersebut terpenuhi. Tahap ini dilakukan sebanyak sepuluh kali perulangan untuk mendapatkan hasil yang lebih akurat, dengan mengambil rata-rata dari waktu eksekusi masing-masing algoritme. Hasil pengujian tahap ini selengkapnya dapat terlihat pada Tabel 4.

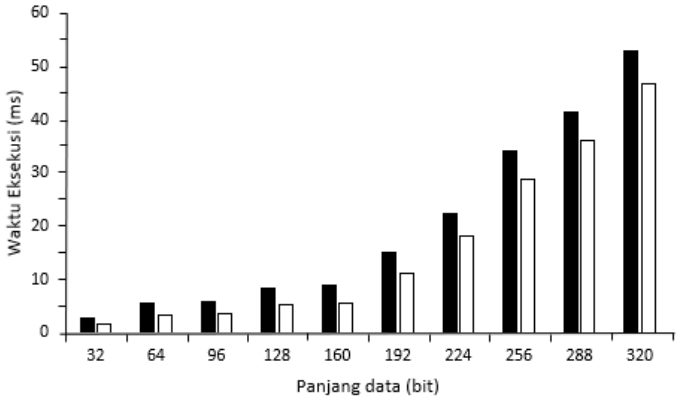
Tabel 4 Hasil pengujian kinerja berdasarkan waktu eksekusinya

Panjang data (bit)	Waktu enkripsi (ms)		Waktu dekripsi (ms)	
	SIMON	SPECK	SIMON	SPECK
32	3.184	2.084	2.736	1.632
64	6.208	3.980	5.488	3.388
96	6.168	3.984	5.812	3.712
128	11.048	11.424	8.476	5.328
160	10.016	11.420	8.792	5.644
192	28.940	28.068	15.200	11.016
224	27.908	28.064	22.480	18.296
256	46.824	45.288	34.076	28.840
288	45.792	45.300	41.356	36.120
320	64.704	63.080	52.948	46.656

Hasil pengujian tersebut menunjukkan bahwa panjang data maksimal yang dapat diproses oleh algoritme SIMON dan SPECK adalah 320 bit atau sebanyak 40 karakter. Untuk membandingkan waktu eksekusi kedua algoritme, hasil tersebut kemudian dibuat menjadi grafik, seperti yang terlihat pada Gambar 12 untuk proses enkripsi dan Gambar 13 untuk proses dekripsi.



Gambar 12 Pengaruh panjang data terhadap waktu eksekusi pada proses enkripsi algoritme SIMON (■) dan SPECK (□) dengan ukuran blok 64 bit dan ukuran kunci 128 bit



Gambar 13 Pengaruh panjang data terhadap waktu eksekusi pada proses dekripsi algoritme SIMON (■) dan SPECK (□) dengan ukuran blok 64 bit dan ukuran kunci 128 bit

Kedua grafik tersebut menunjukkan bahwa panjang data yang diproses mempengaruhi waktu eksekusi program, yakni semakin besar panjang data yang digunakan maka waktu eksekusi program akan semakin lama. Pada proses enkripsi terlihat ada beberapa titik yang waktu eksekusinya tidak berubah, hal ini terjadi karena pada titik tersebut panjang data tidak memenuhi satu blok, sehingga tidak menambah waktu eksekusi. Berdasarkan kedua gambar tersebut, terlihat bahwa algoritme SPECK memiliki kinerja waktu eksekusi yang lebih baik dibandingkan dengan algoritme SIMON.

Tahap berikutnya dilakukan untuk mengukur besar memori yang digunakan oleh masing-masing algoritme. Tahap ini dilakukan dengan cara yang sama seperti pada tahap sebelumnya, tetapi menggunakan panjang data 320 bit dengan data "abcdefghijklmnopqrstuvwxyz0123456789abcd" dan hanya dilakukan sekali tanpa perulangan. Berdasarkan pengujian ini, algoritme SIMON menggunakan 4754 bytes memori untuk proses enkripsi dan 4724 bytes memori untuk proses dekripsi, atau sekitar 14% dari total memori Arduino Uno. Sedangkan algoritme SPECK menggunakan 4540 bytes memori untuk proses enkripsi dan 4492 bytes memori untuk proses dekripsi, atau sekitar 13% dari total memori Arduino Uno. Sehingga dapat disimpulkan bahwa berdasarkan besar memori yang digunakan, algoritme SPECK memiliki kinerja yang lebih baik dibandingkan algoritme SIMON.

Berikutnya tahap terakhir dilakukan untuk mengukur kompleksitas masing-masing algoritme berdasarkan kode program yang terdapat pada Lampiran 3 dan Lampiran 5. Hasil pengujian ini selengkapnya dapat dilihat pada Tabel 5.

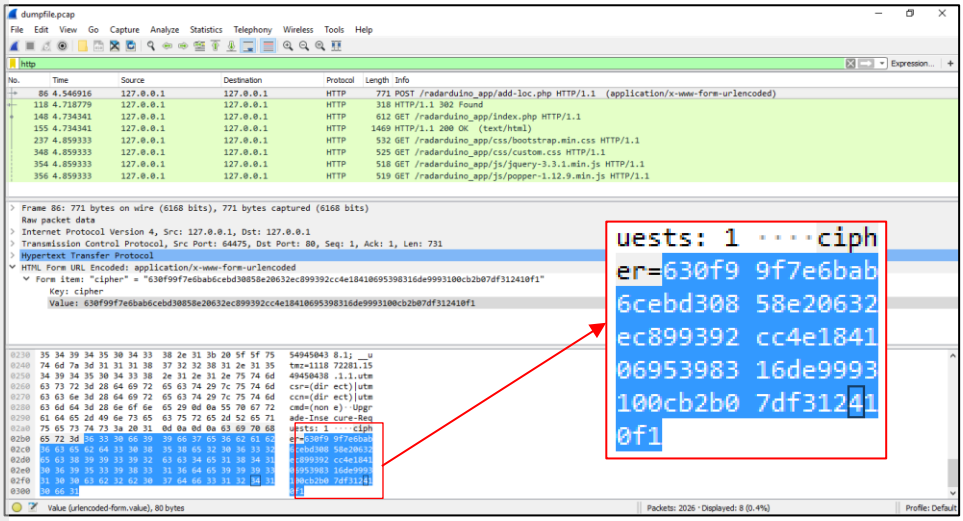
Tabel 5 Hasil pengujian kompleksitas masing-masing algoritme

Bagian	Algoritme SIMON	Algoritme SPECK
Penentuan parameter	4	6
Penjadwalan kunci	$m + 9T$	$1 + m + 5T$
Fungsi ronde	$n(9T)$	$n(5T)$
Total	$9Tn + 9T + m + 4$	$5Tn + 5T + m + 7$
Notasi Big O	$O(n)$	$O(n)$

Tahap ini dilakukan dengan menjumlahkan banyak instruksi mesin yang akan dieksekusi, dan kemudian menyederhanakan ekspresi tersebut ke batas atas kompleksitas (Big O). Pada tabel 5 di atas, terlihat algoritme SIMON memiliki ekspresi total instruksi mesin $9Tn + 9T + m + 4$, sedangkan algoritme SPECK memiliki ekspresi total instruksi mesin $5Tn + 5T + m + 7$, dengan T adalah jumlah ronde, m adalah jumlah *key words*, dan n adalah jumlah blok data yang diproses. Berikutnya, ekspresi tersebut disederhanakan ke dalam notasi Big O. Berdasarkan Tabel 1, maksimum nilai T adalah 72 dan maksimum nilai m adalah 4, maka nilai T, m, dan konstanta tidak terlalu berpengaruh ketika n bernilai sangat besar dan dapat diabaikan dalam notasi Big O. Sehingga dapat disimpulkan algoritme SIMON dan SPECK keduanya memiliki kompleksitas yang sama, yaitu O(n) dengan n adalah jumlah blok data yang diproses. Hal tersebut menunjukkan bahwa algoritme SIMON dan SPECK tidak memerlukan sumber daya banyak dan efisien diterapkan pada perangkat bersumber daya rendah.

3 Pengujian Keamanan

Pengujian ini dilakukan dengan menangkap data yang dikirim oleh *microcontroller* ke laptop, menggunakan aplikasi Wireshark sebagai simulasi *Man-in-the-middle attack*. Hasil Pengujian dapat dilihat pada Gambar 14.



Gambar 14 Hasil pengujian menggunakan aplikasi Wireshark

Dalam pengujian ini, data yang akan dikirim adalah alamat IP perangkat beserta lokasi peragkat saat ini yang berukuran 34 bytes karakter, yaitu "192.168.0.1/-6.6325764/106.8295898" atau jika diubah ke heksadesimal adalah "3139322e3136382e302e312f2d362e363332353736342f3130362e38-323935383938". Sedangkan data yang berhasil didapatkan oleh aplikasi Wireshark adalah "630f99f7e6bab6cebd30858e20632ec899392cc4e1841069-5398316de9993100cb2b07df312410f1" yang berukuran 40 bytes hexadesimal, data tersebut merupakan *cipher* atau data yang telah terenkripsi dan sangat berbeda dengan data aslinya. Sehingga ketika terjadi serangan *Man-in-the-middle attack*, orang yang tidak berwenang tidak dapat mengetahui data aslinya dan tidak akan mempengaruhi keamanan sistem.

SIMPULAN DAN SARAN

Simpulan

Dari hasil percobaan yang dilakukan pada penelitian ini, dapat ditarik beberapa simpulan sebagai berikut:

1 Algoritme SIMON dan SPECK dapat diterapkan pada perangkat IoT untuk menjaga kerahasiaan data.
2 Algoritme SIMON dan SPECK terbukti dapat diterapkan pada perangkat IoT bersumber daya rendah.
3 Panjang data yang diolah mempengaruhi waktu eksekusi algoritme. Semakin besar panjang data yang digunakan, maka semakin lama waktu eksekusinya.
4 Berdasarkan waktu eksekusi dan besar memori yang digunakan, algoritme SPECK memiliki kinerja yang lebih baik dibandingkan algoritme SIMON pada perangkat Arduino Uno.
5 Simulasi sistem dapat berjalan dengan baik di luar ruangan, tetapi tidak di dalam ruangan karena keterbatasan perangkat GPS.

Saran

Saran yang dapat diberikan untuk penelitian lebih lanjut adalah sebagai berikut:

1 Menerapkan prototipe sistem yang ada pada dunia nyata agar mendapat hasil yang lebih objektif.
2 Menerapkan jenis layanan keamanan lain untuk mencapai tujuan keamanan informasi lainnya seperti integritas data, autentikasi dan non-repudiasi.
3 Menguji prototipe sistem terhadap berbagai serangan yang mungkin terjadi untuk mengetahui seberapa kuat sistem yang telah dibuat.

DAFTAR PUSTAKA

Babar S, Stango A, Prasad N, Sen J, Prasad R. 2011. Proposed embedded security framework for internet of things (IoT). Di dalam: *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology* (*Wireless VITAE*). 2011 Feb 28-Mar 3. Chennai (IN). hlm 1-5.

Beaulieu R, Treatman-Clark S, Shors D, Weeks B, Smith J, Wingers L. The SIMON and SPECK lightweight block ciphers. Di dalam: *2015 52nd ACM/EDAC/IEEE Design Automation Conference* (*DAC*). 2015 Jun 8. San Francisco (US): IEEE. hlm 1-6.

Dinu D, Corre YL, Khovratovich D, Perrin L, Großschädl J, Biryukov A. 2015. Triathlon of lightweight block ciphers for the internet of things. *Journal of Cryptographic Engineering*. 5(1): 1-20.

Hossain M, Hasan R, Skjellum A. 2017. Securing the internet of things: a meta-study of challenges, approaches, and open problems. Di dalam: *2017 IEEE 37th*

International Conference on Distributed Computing Systems Workshops (ICDCSW). 2015 Jun 5-8. Atlanta, GA (US): IEEE. hlm 220-225.

Jing Q, Vasilakos AV, Wan J. Lu J, Qiu D. 2014. Security of the internet of things: perspectives and challenges. *Wireless Network.* 20(8): 2481-2501.

Keerthana M, Parveen SA. 2017. Internet of things. *International Journal of Advanced Research Methodology in Engineering & Technology*. 1(2): 105-108.

Li S, Xu LD. 2017. *Securing the Internet of Things*. Cambridge (US): Syngress.

Menezes AJ, Oorschot PC, dan Vanstone SA. 1996. *Handbook of Applied Cryptography*. New York (US): CRC.

Stankovic JA. 2014. Research directions for the internet of things. *IEEE Internet of Things Journal*. 1(1): 3-9.

Lampiran 1 Contoh hasil *tracing* algoritme SIMON dengan ukuran blok 64 bit dan ukuran kunci 128 bit

Data yang diposes:
"abcdefgh" → x = "abcd"; y = "efgh";

Kunci yang digunakan:
"0123456789abcdef" → key[0] = "0123"; key[1] = "4567";
key[2] = "89ab"; key[3] = "cdef";

Penentuan parameter:
- Jumlah *key words* = $m = 4$
- Urutan konstanta SIMON yang digunakan = $z_j = 3$
- *Bit mask* = $c = 3$
- Jumlah ronde = 44

Penjadwalan kunci:
k[0] = key[m-1] = key[3] = "cdef" = 0x63646566
k[1] = key[m-2] = key[2] = "89ab" = 0x38396162
k[2] = key[m-3] = key[1] = "4567" = 0x34353637
k[3] = key[m-4] = key[0] = "0123" = 0x30313233
k[4] = $c \oplus$ Simonz[z_j][i-m mod 62] $\oplus$ k[i-m] $\oplus$ ($I \oplus S^{-1}$) (S^{-3} k[i-1] $\oplus$ k[i-3])
= 3 $\oplus$ Simonz[3][4-4 mod 62] $\oplus$ k[4-4] $\oplus$ ($I \oplus S^{-1}$) (S^{-3} k[4-1] $\oplus$ k[4-3])
= 3 $\oplus$ 1 $\oplus$ 0x63646566 $\oplus$ ($I \oplus S^{-1}$) (S^{-3} (0x30313233) $\oplus$ 0x38396162)
= 0xedbb7e2d
k[5] = 3 $\oplus$ Simonz[3][5-4 mod 62] $\oplus$ k[5-4] $\oplus$ ($I \oplus S^{-1}$) (S^{-3} k[5-1] $\oplus$ k[5-3])
= 3 $\oplus$ 1 $\oplus$ 0x38396162 $\oplus$ ($I \oplus S^{-1}$) (S^{-3} (0xedbb7e2d) $\oplus$ 0x34353637)
= 0x0a85eb94
⋮
k[43] = 3 $\oplus$ Simonz[3][43-4 mod 62] $\oplus$ k[43-4] $\oplus$ ($I \oplus S^{-1}$)(S^{-3} k[43-1] $\oplus$ k[43-3])
= 3 $\oplus$ 0 $\oplus$ 0xf6f19d46 $\oplus$ ($I \oplus S^{-1}$) (S^{-3} (0xaf237791) $\oplus$ 0x03a705ac)
= 0x246cbc4b

Fungsi ronde enkripsi:
$ronde_0$ → $tmp = x$ = 0x61626364
$x = (S^1 tmp \ \& \ S^8 tmp) \oplus y$
= (S^1(0x61626364) & S^8(0x61626364)) $\oplus$ 0x65666768
= 0x27262328
$x = S^2\ tmp \oplus x$
= S^2 (0x61626364) $\oplus$ 0x27262328
= 0xa2afaeb9
x = k[0] $\oplus$ x
= 0x63646566 $\oplus$ 0xa2afaeb9
= 0xc1cbcbdf
y = tmp = 0x61626364
⋮

Lampiran 1 Lanjutan

$ronde_{43}$ → $tmp = x =$ 0xbe2f342a

$x = (S^1 tmp \ \& \ S^8 tmp) \oplus y$

$= (S^1(\text{0xbe2f342a}) \ \& \ S^8(\text{0xbe2f342a})) \oplus \text{0x0a398830}$

$= \text{0x262da024}$

$x = S^2 tmp \oplus x$

$= S^2(\text{0xbe2f342a}) \oplus \text{0x262da024}$

$= \text{0xde91708e}$

$x = \text{k[43]} \oplus x$

$= \text{0x246cbc4b} \oplus \text{0xde91708e}$

$= \text{0xfafdccc5}$

$y = tmp = \text{0xbe2f342a}$

Cipher:

"fafdccc5be2f342a"

Fungsi ronde dekripsi:

$ronde_0$ → $tmp = y =$ 0xbe2f342a

$y = (S^1 tmp \ \& \ S^8 tmp) \oplus x$

$= (S^1(\text{0xbe2f342a}) \ \& \ S^8(\text{0xbe2f342a})) \oplus \text{0xfafdccc5}$

$= \text{0xd6e9e4d1}$

$y = S^2 tmp \oplus y$

$= S^2(\text{0xbe2f342a}) \oplus \text{0xd6e9e4d1}$

$= \text{0x2e55347b}$

$y = \text{k[44-0-1]} \oplus y$

$= \text{0x246cbc4b} \oplus \text{0x2e55347b}$

$= \text{0x0a398830}$

$x = tmp = \text{0xbe2f342a}$

⋮

$ronde_{43}$ → $tmp = y =$ 0x61626364

$y = (S^1 tmp \ \& \ S^8 tmp) \oplus x$

$= (S^1(\text{0x61626364}) \ \& \ S^8(\text{0x61626364})) \oplus \text{0xc1cbcbdf}$

$= \text{0x838b8f9f}$

$y = S^2 tmp \oplus y$

$= S^2(\text{0x61626364}) \oplus \text{0x838b8f9f}$

$= \text{0x0602020e}$

$y = \text{k[44-43-1]} \oplus y$

$= \text{0x63646566} \oplus \text{0x0602020e}$

$= \text{0x65666768}$

$x = tmp = \text{0x61626364}$

Plaintext:

0x61626364 0x65666768 = "abcdefgh"

Lampiran 2 Contoh hasil *tracing* algoritme SPECK dengan ukuran blok 64 bit dan ukuran kunci 128 bit

Data yang diposes:

"abcdefgh" → x = "abcd"; y = "efgh";

Kunci yang digunakan:

"0123456789abcdef" → key[0] = "0123"; key[1] = "4567";
key[2] = "89ab"; key[3] = "cdef";

Penentuan parameter:

- Jumlah *key words* = $m = 4$
- Jumlah rotasi $\alpha = 8$
- Jumlah rotasi $\beta = 3$
- Jumlah ronde = 27

Penjadwalan kunci:

k[0] = key[m-1] = key[3] = "cdef" = 0x63646566
l[0] = key[m-2] = key[2] = "89ab" = 0x38396162
l[1] = key[m-3] = key[1] = "4567" = 0x34353637
l[2] = key[m-2] = key[0] = "0123" = 0x30313233

ronde$_0$ → l[0+4-1] = (k[0] + S^{-8} l[0]) ⊕ 0
l[3] = (0x63646566 + S^{-8} (0x38396162)) ⊕ 0
= 0xc59c9ec7
k[0+1] = S^3 k[0] ⊕ l[0+4-1]
k[1] = S^3 (0x63646566) ⊕ 0xc59c9ec7
= 0xdebfb5f4
⋮

ronde$_{25}$ → l[25+4-1] = (k[25] + S^{-8} l[25]) ⊕ 25
l[28] = (0x03a199d7+ S^{-8} (0x3a745f33)) ⊕ 25
= 0x36dc0e2f
k[25+1]= S^3 k[25] ⊕ l[25+4-1]
k[26] = S^3 (0x03a199d7) ⊕ 0x36dc0e2f
= 0x2bd0c097

Fungsi ronde enkripsi:

ronde$_0$ → x = ($S^{-\alpha}$ x + y) ⊕ k[0]
= (S^{-8}(0x61626364) + 0x65666768) ⊕ 0x63646566
= 0xaaa3acad
y = S^{β} y ⊕ x
= S^3 (0x65666768) ⊕ 0xaaa3acad
= 0x819097ee
⋮

ronde$_{26}$ → x = ($S^{-\alpha}$ x + y) ⊕ k[26]
= (S^{-8}(0x669c35e0) + 0xb563bfaa) ⊕ 0x2bd0c097
= 0xbe1a9b48

Lampiran 2 Lanjutan

$$y = S^{\beta}\, y \oplus x$$
$$= S^{3}\,(0xb563bfaa) \oplus 0xbe1a9b48$$
$$= 0x1507661d$$

Cipher:

"be1a9b481507661d"

Fungsi ronde dekripsi:

$ronde_0 \rightarrow$
$$y = S^{-\beta}\,(x \oplus y)$$
$$= S^{-3}\,(0xbe1a9b48 \oplus 0x1507661d)$$
$$= 0xb563bfaa$$
$$x = S^{\alpha}\,((x \oplus k[27\text{-}0\text{-}1]) - y)$$
$$= S^{8}\,((0xbe1a9b48 \oplus 0x2bd0c097) - 0xb563bfaa)$$
$$= 0x669c35e0$$

$$\vdots$$

$ronde_{26} \rightarrow$
$$y = S^{-\beta}\,(x \oplus y)$$
$$= S^{-3}\,(0xaaa3acad \oplus 0x819097ee)$$
$$= 0x65666768$$
$$x = S^{\alpha}\,((x \oplus k[27\text{-}26\text{-}1]) - y)$$
$$= S^{8}\,((0xaaa3acad \oplus 0x63646566) - 0x65666768)$$
$$= 0x61626364$$

Plaintext:

0x61626364 0x65666768 = "abcdefgh"

Lampiran 3 Kode program enkripsi algoritme SIMON dengan ukuran blok 64 bit dan ukuran kunci 128 bit

```
// Circular shift
#define ROTL( n, X ) ( ( ( X ) << n ) | ( ( X ) >> ( 32 - n ) ) )
#define ROTR( n, X ) ( ( ( X ) >> n ) | ( ( X ) << ( 32 - n ) ) )

// Struct untuk menyimpan ciphertext
struct Text {
   uint32_t left;
   uint32_t right;
};

// Konstanta SIMON
char Simonz[5][62] = {
       "11111010001001010110000111001101111101000100101011000011100110",
       "10001110111110010011000010110101000111011111100100110000101011010",
       "10101111011100000011010010011000101000010001111110010110110011",
       "11011011101011000110010111100000010010001010011100110100001111",
       "11010001111001101011011000100000010111000011001010010011101111"
};

// Fungsi enkripsi algoritme SIMON
Text SimonEncrypt64_128(uint32_t PL, uint32_t PR, uint32_t* key)
{
  int mm=4, T=44, Zj=3;
  uint32_t k[44]={0};
  Text ciphertext;
  int i,j;

  // ------------------------ penjadwalan kunci ------------------------
  for (i=0; i<mm; i++)
  {
    k[i]= key[i];
  };
  for (i=mm; i<T; i++)
  {
    uint32_t tmp = ROTR(3, k[i-1]);
    if (mm == 4) tmp ^= k[i-3];
    tmp = ~(tmp) ^ ROTR(1, tmp);
    k[i] = k[i-mm] ^ tmp ^ (Simonz[Zj][(i-mm) % 62]-'0') ^ 3;
  };

  // ---------------------------- enkripsi ----------------------------
  for (i=0; i<T; i++)
  {
    uint32_t tmp = PL;
    PL = PR ^ ROTL(1, PL) & ROTL(8, PL) ^ ROTL(2, PL) ^ k[i];
    PR = tmp;
  };
  ciphertext.left = PL;
  ciphertext.right = PR;

  return ciphertext;
}
```

Lampiran 4 Kode program dekripsi algoritme SIMON dengan ukuran blok 64 bit dan ukuran kunci 128 bit

```
// Circular shift
#define ROTL( n, X ) ( ( ( X ) << n ) | ( ( X ) >> ( 32 - n ) ) )
#define ROTR( n, X ) ( ( ( X ) >> n ) | ( ( X ) << ( 32 - n ) ) )

// Struct untuk menyimpan plaintext
struct Text {
   uint32_t left;
   uint32_t right;
};

// Konstanta SIMON
char Simonz[5][62] = {
     "11111010001001010110000111001101111101000100101011000011100110",
     "10001110111110010011000010110101000111011111001001100001011010",
     "10101111011100000011010010011000101000010001111110010110110011",
     "11011011101011000110010111100000010010001010011100110100001111",
     "11010001111001101011011000100000010111000011001010010011101111"
};

// Fungsi dekripsi algoritme SIMON
Text SimonDecrypt64_128(uint32_t CL, uint32_t CR, uint32_t* key)
{
  int mm=4, T=44, Zj=3;
  uint32_t k[44]={0};
  Text plaintext;
  int i, j;

  // ------------------------ penjadwalan kunci ------------------------
  for (i=0; i<mm; i++)
  {
    k[i] = key[i];
  };
  for (i=mm; i<T; i++)
  {
    uint32_t tmp = ROTR(3, k[i-1]);
    if (mm == 4) tmp ^= k[i-3];
    tmp = ~(tmp) ^ ROTR(1, tmp);
    k[i] = k[i-mm] ^ tmp ^ (Simonz[Zj][(i-mm) % 62]-'0') ^ 3;
  };

  // ---------------------------- dekripsi ----------------------------
  for (i=0; i<T; ++i)
  {
    uint32_t tmp = CR;
    CR = CL ^ ROTL(1, CR) & ROTL(8, CR) ^ ROTL(2, CR) ^ k[T-i-1];
    CL = tmp;
  };
  plaintext.left = CL;
  plaintext.right = CR;

  return plaintext;
}
```

Lampiran 5 Kode program enkripsi algoritme SPECK dengan ukuran blok 64 bit dan ukuran kunci 128 bit

```
// Circular shift
#define ROTL( n, X ) ( ( ( X ) << n ) | ( ( X ) >> ( 32 - n ) ) )
#define ROTR( n, X ) ( ( ( X ) >> n ) | ( ( X ) << ( 32 - n ) ) )

// Struct untuk menyimpan ciphertext
struct Text {
   uint32_t left;
   uint32_t right;
};

// Fungsi enkripsi algoritme SPECK
Text SpeckEncrypt64_128(uint32_t PL, uint32_t PR, uint32_t* key)
{
  int mm=4, T=27, alpha=8, beta=3;
  uint32_t k[29]={0};
  uint32_t l[29]={0};
  Text ciphertext;
  int i, j;

  // ----------------------- penjadwalan kunci ------------------------
  k[0] = key[0];
  for (i=0; i<mm-1; i++)
  {
    l[i] = key[i+1];
  };
  for (i=0; i<T-1; i++)
  {
    l[i+mm-1] = (k[i] + ROTR(alpha, l[i])) ^ i;
    k[i+1] = ROTL(beta, k[i]) ^ l[i+mm-1];
  };

  // ---------------------------- enkripsi -----------------------------
  for (i=0; i<T; i++)
  {
    PL = (ROTR(alpha, PL) + PR) ^ k[i];
    PR = ROTL(beta, PR) ^ PL;
  };
  ciphertext.left = PL;
  ciphertext.right = PR;

  return ciphertext;
}
```

Lampiran 6 Kode program dekripsi algoritme SPECK dengan ukuran blok 64 bit dan ukuran kunci 128 bit

```
// Circular shift
#define ROTL( n, X ) ( ( ( X ) << n ) | ( ( X ) >> ( 32 - n ) ) )
#define ROTR( n, X ) ( ( ( X ) >> n ) | ( ( X ) << ( 32 - n ) ) )

// Struct untuk menyimpan plaintext
struct Text {
   uint32_t left;
   uint32_t right;
};

// Fungsi dekripsi algoritme SPECK
Text SpeckDecrypt64_128(uint32_t CL, uint32_t CR, uint32_t* key)
{
  int mm=4, T=27, alpha=8, beta=3;
  uint32_t k[29]={0};
  uint32_t l[29]={0};
  Text plaintext;
  int i, j;

  // ------------------------ penjadwalan kunci ------------------------
  k[0] = key[0];
  for (i=0; i<mm-1; i++)
  {
    l[i] = key[i+1];
  };
  for (i=0; i<T-1; i++)
  {
    l[i+mm-1] = (k[i] + ROTR(alpha, l[i])) ^ i;
    k[i+1] = ROTL(beta, k[i]) ^ l[i+mm-1];
  }

  // ---------------------------- dekripsi ----------------------------
  for (i=0; i<T; ++i)
  {
    CR = ROTR(beta, (CL ^ CR));
    CL = ROTL(alpha, ((CL ^ k[T-i-1]) - CR));
  };
  plaintext.left = CL;
  plaintext.right = CR;

  return plaintext;
}
```

RIWAYAT HIDUP

Penulis yang bernama lengkap Wawan Setyadi dilahirkan di kota Bogor, 8 Juli 1997. Penulis merupakan anak tunggal dari pasangan Kusnanto dan Dra Tri Komiyati. Penulis menyelesaikan pendidikan tingkat SMP pada tahun 2012 di SMP Negeri 7 Bogor dan tingkat SMA pada tahun 2014 di SMA Negeri 3 Bogor. Pada tahun 2014, penulis diterima sebagai mahasiswa Departemen Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor melalui jalur SBMPTN.

Semasa menyandang status sebagai mahasiswa, penulis aktif terlibat dalam komunitas *Web Developer* IPB serta aktif mengikuti kompetisi desain dan pengembangan sistem informasi berskala nasional pada tahun 2016 hingga 2018. Dari kompetisi tersebut, penghargaan yang pernah diterima penulis adalah juara 2 lomba desain web pada acara Confest 2016 di Universitas Bangka Belitung. Penulis juga pernah melaksanakan Praktek Kerja Lapang selama dua bulan pada tahun 2017 di PT. Media Indonusa sebagai *system developer*.