Analisis Perbandingan Block Cipher Simon-Speck, Simeck, dan Skinny pada Komunikasi Berbasis LoRa

Winda Ezranata Putri[1], Favian Dewanta[1], Farah Afianti[2]
[1]Teknik Telekomunikasi, Fakultas Teknik Elektro, Telkom University
Jl. Telekomunikasi No. 1, Bandung, 0227566456
[2]Teknik Informatika, Fakultas Teknik Informatika, Telkom University
Jl. Telekomunikasi No. 1, Bandung, 0227566456
windaezranata@student.telkomuniversity.ac.id, favian@telkomuniversity.ac.id, farahafi@telkomuniversity.ac.id

Diterima: 2 September 2022. Disetujui: 3 November 2022. Dipublikasikan: 17 November 2022.

Abstract - Inventions solving problems through the incorporation of technology and social impacts to facilitate human life can be defined as the Internet of Things (IoT). Security and privacy are challenges in various Internet of Things applications given the rapid growth of IoT devices. Meanwhile, IoT devices still do not have sufficient communication security. Communication systems on IoT devices require cryptographic to maintain the security of their communications. The appropriate algorithm to be implemented on IoT devices should be the smallest and fastest lightweight cryptographic algorithm or commonly called Lightweight Cryptography because IoT devices have limited memory and computing power. In this study, the proposed algorithm to be implemented in Long Range or LoRa-based IoT devices is the Simon-Speck, Simeck, and Skinny encryption algorithm. This study discusses the implementation of the Simon-Speck, Simeck, and Skinny algorithms on Long Range-based IoT devices. The parameters compared are the computation time of the encryption and decryption process and the avalanche effect value. In the encryption and decryption process, the Speck algorithm has the fastest computation time. The results showed that the fastest encryption and decryption times were obtained at 0.003 seconds and 0.004 seconds. Furthermore, the highest avalanche effect value is obtained by the Skinny algorithm, with a value of 50.34%.

Keywords: cryptography, Simon-Speck, Simeck, Skinny, LoRa

Abstrak - Penemuan yang mampu menyelesaikan permasalahan yang ada melalui penggabungan teknologi dan dampak sosial agar dapat memudahkan kehidupan manusia dapat didefinisikan sebagai Internet of Things (IoT). Keamanan dan privasi menjadi tantangan di berbagai pengaplikasian Internet of Things mengingat semakin cepatnya pertumbuhan perangkat IoT. Sedangkan, perangkat IoT masih belum memiliki keamanan komunikasi yang cukup. Sistem komunikasi pada perangkat IoT memerlukan algoritma kriptografi dan menerapkan sistem enkripsi dan dekripsi untuk menjaga keamanan komunikasinya. Algoritma yang sesuai untuk diimplementasikan pada perangkat IoT sebaiknya merupakan algoritma kriptografi ringan yang terkecil dan tercepat atau biasa disebut Lightweight Cryptography karena perangkat IoT memiliki keterbatasan memori dan daya komputasi. Dalam penelitian ini, algoritma yang diajukan untuk diimplementasikan pada alat IoT berbasis Long Range atau LoRa adalah algoritma enkripsi Simon-Speck, Simeck, dan Skinny. Penelitian ini membahas tentang implementasi algoritma Simon-Speck, Simeck, dan Skinny pada perangkat IoT berbasis Long Range. Parameter yang dibandingkan adalah waktu komputasi proses enkripsi dan dekripsi serta nilai avalanche effect. Pada proses enkripsi dan dekripsi, algoritma Speck memiliki waktu komputasi paling cepat. Hasil penelitian menunjukkan bahwa waktu enkripsi dan dekripsi paling cepat diperoleh sebesar 0,003 detik dan 0,004 detik. Selanjutnya, nilai avalanche effect paling tinggi diperoleh algoritma Skinny, dengan nilai sebesar 50,34%.

Kata kunci: kriptografi, Simon-Speck, Simeck, Skinny, LoRa

I. PENDAHULUAN

Teknologi informasi yang berkembang saat ini membuat dunia terhubung secara digital. Teknologi yang sangat maju juga sudah dirasakan oleh banyak orang dan mempengaruhi banyak sektor dan bidang kehidupan. Perkembangan teknologi informasi membuat aktivitas manusia menjadi semakin mudah dan cepat. Hal tersebut membuat teknologi di Indonesia berkembang dengan pesat. Di era Revolusi Industri 4.0, tren ini mengintegrasikan teknologi siber (*cyber*) dan teknologi otomatisasi (*automation)*. Praktik industri dan manufaktur tradisional digabungkan dengan teknologi-teknologi yang inovatif, seperti *Machine-to-Machine* (M2M), IoT, *Cyber Physical Systems* (CPS), dll pada era Revolusi Industri 4.0 [1].

Salah satu inovasi yang menjadi pilar terbesar di era Revolusi Industri 4.0 adalah *Internet of Things* (IoT). IoT memiliki kemampuan dalam menyambungkan dan memudahkan proses komunikasi antara mesin, perangkat, sensor, dan manusia melalui jaringan internet [2]. IoT berkontribusi secara signifikan untuk meningkatkan kehidupan kita sehari-hari di berbagai aplikasi dan

berbagai macam sektor. Perangkat IoT dapat mengumpulkan dan menerima informasi dari internet dalam keadaan berpindah-pindah sekalipun. Penerapan dan penelitian di bidang IoT telah mengalami pertumbuhan dan perkembangan yang sangat cepat dalam beberapa tahun terakhir. Dari tahun 2006 sampai 2018, terdapat 8510 jurnal publikasi dan 16775 makalah konferensi *proceeding* yang membahas tentang IoT [3][4]. Bukti pendukung lainnya adalah menurut laporan dari Ericsson, di tahun 2021 terdapat sekitar 28 miliar perangkat pintar yang terkoneksi di seluruh dunia [3].

Perangkat IoT yang sangat berkembang beberapa tahun belakang ini membuat keamanan dan privasi menjadi tantangan pada pengaplikasian perangkat. Keamanan dan privasi data yang dikumpulkan dan dipertukarkan dapat menjadi isu yang berpotensi mengancam perangkat IoT. Data sering kali berkaitan dengan identitas pribadi pengguna, sehingga keamanan dan privasi data sangat penting untuk dilindungi. Keamanan data diperlukan agar informasi pribadi dapat dengan tidak mudah dicuri jika sewaktu-waktu sistem diretas. Perangkat IoT yang tidak aman menjadi perhatian nyata mengingat lebih dari 25% dari serangan siber melibatkan perangkat IoT [5]. Penerapan sistem keamanan dalam aplikasi IoT sangat penting untuk mengurangi risiko IoT.

Pengaplikasian protokol keamanan dibutuhkan untuk mengamankan data yang ditransfer antar perangkat. Salah satu caranya ialah penerapan algoritma enkripsi pada komunikasi data. Data dapat dienkripsi menjadi teks yang berbeda dan tidak mudah dipahami karena adanya kriptografi. Algoritma yang cocok digunakan adalah algoritma kriptografi yang bersifat ringan atau *Lightweight Cryptographic Algorithms* (LWC) karena LWC dapat mengurangi kompleksitas komputasi tetapi tetap mempertahankan tingkat keamanan [6-8].

Hussain menggunakan metode enkripsi yang terautentikasi untuk mencapai aspek *confidentiality* dan *integrity* pada IEC 61850 GOOSE *message* [9]. Penelitian ini mengusulkan tiga metode untuk yaitu Encrypt-then-MAC (EtM), Encrypt-and-MAC (E&M), dan MAC-then-Encrypt (MtE). Metode enkripsi dijalankan menggunakan algoritma AES 128 dan AES 256 untuk mengamankan aspeck *confidentiality*. Algoritma MAC digunakan pada metode autentikasi pesan. Percobaan laboratorium menyimpulkan bahwa algoritma AEAD dapat berhasil diterapkan ke pesan GOOSE sambil memenuhi persyaratan *delay* 3 ms yang ketat.

Penelitian yang dilakukan oleh Iyer dkk. membahas tentang implementasi dan evaluasi dari algoritma enkripsi ringan pada lingkungan MQTT [10]. Enkripsi data/*payload* berguna untuk mengamankan data sensitif yang dikirim oleh *publisher* dari pihak yang tidak berwenang. Payload MQTT dienkripsi menggunakan algoritma enkripsi ringan Simon-Speck, Present, Pride, Roadrunner, Midori, dan Rectangle. Penelitian ini mengamati performa dari semua algoritma simetris tersebut saat diaplikasikan pada perangkat IoT, yang mana algoritma Simon-Speck memiliki waktu eksekusi yang lebih baik dibandingkan algoritma lain. Di tahun 2020, penelitian senada dilakukan oleh Yustiarini dkk. yang membandingkan Simon-Speck dengan AES pada komunikasi perangkat IoT berbasiskan MQTT [11]. Hasil eksperimen menunjukkan bahwa algoritma Speck memiliki performa jaringan terbaik (delay dan throughput) serta penggunaan memori yang rendah jika dibandingkan dengan algoritma AES dan Simon. Namun dalam hal avalanche effect, algoritma Simon memiliki performa terbaik jika dibandingkan dengan algoritma AES dan Speck. Penelitian ini semakin menegaskan bahwa algoritma Simon-Speck merupakan algoritma ringan dan aman jika dibandingkan dengan algoritma block cipher yang sudah mapan digunakan seperti AES.

Penelitian yang mengimplementasikan *block cipher* Simeck pada MSP430 dilakukan oleh Taewan, dkk. Penulis mengusulkan metode implementasi yang efisien menurut Instrumen Texas MSP430 untuk ukuran kode dan optimasi kecepatan. Algoritma enkripsi Simeck diimplementasikan pada mikrokontroler 16-bit MSP430G2553. Penulis mengekstrak ekspansi kunci dan proses enkripsi dari referensi yang dibuat oleh Bo Zhu agar mencapai implemetasi yang efisien, lalu mengukur performansinya [12]. Penggunaan ROM dan RAM serta pengukuran siklus waktu merupakan sesuatu yang dievaluasi pada penelitian ini. Hal yang dicapai penulis pada penelitian ini adalah ukuran kode yang lebih kecil dan waktu enkripsi yang lebih cepat.

Implementasi algoritma Skinny pada FPGA yang hemat area dan cocok untuk aplikasi IoT yang ringan dibahas pada penelitian yang dilakukan oleh Xiang Feng dan Shugou Li. Implementasi yang diajukan adalah *column-serial structure*. Lima operasi pada Skinny dikelompokkan menjadi tiga tahap. Tahap pertama menampilkan Subcells, AddConstant, dan AddRoundTweakey secara kolom demi kolom dan membutuhkan 5 *clock cycle*. Tahap kedua menapilkan operasi ShiftRows dalam satu *clock cycle*. Tahap ketiga pada operasi MixColumns membutuhkan empat *clock cycle*. Secara total, *column-serial structure* hanya membutuhkan 10 *clock cycle* untuk satu putaran proses enkripsi. Penulis melakukan implementasi ASIC pencipta Skinny ke platform FPGA yang sama dengan milik

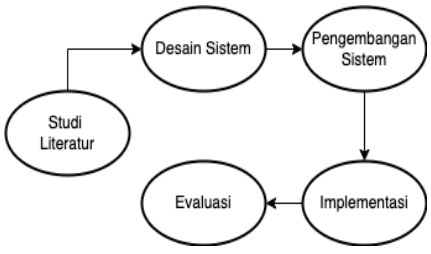
penulis. Hasil perbandingan menunjukkan bahwa metode *column-serial* mengalami 88% pengurangan LUT (*lookup table*) jika dibandingkan dengan metode *round-based*. Perbandingan yang kedua mencapai 220% lebih cepat jika dibandingkan dengan implementasi *bible-serial.* Indikator waktu pengali sumber daya menunjukkan bahwa metode *column-serial* mencapai efisiensi area terbaik.

Pada tahun 2014, Badan Nasional Amerika Serikat mengusulkan algoritma Simon-Speck dalam standar kriptografi ringan ISO/IEC SC27 WG2, ISO/IEC 29192-2 [13]. Algoritma Skinny diciptakan untuk bisa bersaing dengan algoritma Simon [14]. Selain itu, algoritma Simeck merupakan kombinasi dari algoritma Simon-Speck. Berdasarkan beberapa penelitian sebelumnya, keempat algoritma *block cipher*, yaitu Simon-Speck, Simeck dan Skinny belum pernah dibandingkan. Lalu, algoritma Simon-Speck, Simeck, dan Skinny juga belum ada pengimplementasiannya pada protokol komunikasi LoRa atau *Long Range*. Oleh karena itu, mengimplementasikan keempat algoritma tersebut pada protokol komunikasi LoRa menjadi pembahasan yang baru untuk diteliti.

Hal yang diamati dari penelitian ini adalah menguji keempat algoritma dan membandingkan waktu komputasinya serta mengukur dan membandingkan nilai *avalanche effect* antar keempat algoritma. Algoritma Simon-Speck, Simeck, dan Skinny diimplementasikan pada perangkat IoT menggunakan protokol komunikasi LoRa. Hasil dari pengujian ini akan diperoleh algoritma terbaik dari segi waktu maupun performansi, yang nantinya akan digunakan untuk perangkat IoT.

II. METODE PENELITIAN

Metode penelitian yang digunakan pada penelitian ini adalah metode kuantitatif, di mana beberapa pengujian perlu dilakukan untuk menghasilkan suatu keluaran yang dapat dipercaya. Langkah-langkah yang dilakukan dalam metode penelitian ditampilkan pada Gambar 1.



Gambar 1. Alur Proses Metode Penelitian

A. Studi Literatur

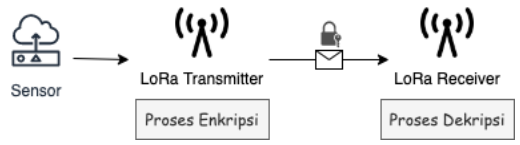
Analisis terhadap perancangan desain sistem dilakukan dengan mempelajari algoritma enkripsi Simon-Speck, Simeck, dan Skinny dengan membaca beberapa literatur yang berisi tentang algoritma enkripsi Simon-Speck, Simeck, dan Skinny [10][11][15]. Beberapa literatur menyebutkan bahwa algoritma Simon-Speck merupakan algoritma kriptografi ringan yang terkecil dan tercepat pada semua *platform*. Hal ini membuat algoritma Simon-Speck cocok diimplementasikan pada perangkat IoT. Kombinasi dari algoritma Simon-Speck, yaitu Simeck juga akan diimplementasikan dan dianalisis untuk menentukan algoritma apa yang lebih baik untuk keamanan pada alat IoT. Penambahan algoritma yaitu algoritma Skinny juga menjadi pilihan untuk menganalisis algoritma enkripsi yang cocok untuk diterapkan pada perangkat IoT. Keempat algoritma tersebut beroperasi dalam blok data.

B. Desain Sistem

Desain dari penelitian ini menggunakan algoritma Simon-Speck, Simeck, dan Skinny. Perancangan blok sistem diperlukan untuk gambaran pengimplementasian alat. Blok sistem terdiri dari tiga proses, yaitu sensing pada sensor, proses enkripsi pada transmitter, proses dekripsi pada receiver. Gambar dari blok sistem dapat dilihat pada Gambar 2. Proses pertama merupakan proses sensing yang dilakukan oleh sensor. Pesan yang harus diubah menjadi ciphertext didapatkan dari data-data sensor. Data yang telah diperoleh dari beberapa sensor akan digabungkan menjadi satu pesan utuh. Pesan tersebut yang menjadi input plaintext di setiap algoritma enkripsi.

Proses enkripsi dilakukan pada transmitter. Transmitter berupa Raspberry Pi dan algoritma enkripsi menggunakan bahasa pemrograman Python. Algoritma yang digunakan untuk pengujian terdiri dari empat *block cipher*, yaitu Simon, Speck, Simeck, dan Skinny. Pesan atau *plaintext* akan dienkripsi menggunakan keempat *block cipher*. Hasil dari proses enkripsi merupakan pesan yang sudah acak dan tidak dapat dipahami otak manusia atau dapat disebut *ciphertext*. Setelah pesan berhasil terenkripsi, pesan dikirim dari *transmitter* ke *receiver* menggunakan protokol LoRa.

Pesan yang telah sampai di *receiver* mengalami proses dekripsi. Proses pengembalian pesan acak menjadi pesan awal kembali dilakukan pada *receiver* yang berupa Raspberry Pi dan menggunakan bahasa pemrograman Python. Pesan didekripsi menggunakan algoritma yang digunakan pada proses enkripsi.



Gambar 2. Blok Sistem Perancangan Enkripsi

C. Pengembangan Sistem

Pada tahap pengembangan desain sistem, yang dilakukan adalah menyusun *source code* untuk proses enkripsi dan dekripsi. *Source code* yang digunakan adalah *source code* yang memiliki cara kerja yang sama dengan proses *round function* dan *key expansion* sesuai referensi yang telah dipelajari. Diagram alur masing-masing algoritma disusun agar menambah pemahaman terhadap cara kerja algoritma.

D. Implementasi

Setelah *source code* berhasil dibuat, penulis melakukan unggah *source code* ke Raspberry Pi. *Source code* yang berisi algoritma proses enkripsi akan diaplikasikan pada LoRa *Transmitter* dan *source code* untuk proses dekripsi akan diaplikasikan pada LoRa *Receiver*. Pada tahap ini berkas yang algoritma akan dipanggil pada *source code* utama.

E. Evaluasi

Setelah proses enkripsi dan dekripsi berhasil dijalankan pada perangkat, maka penulis akan mengevaluasi performansinya sesuai parameter yang diperlukan. Penulis akan mengamati waktu komputasi yang diperlukan alat IoT untuk melakukan proses enkripsi dan dekripsi. Selain itu, penulis juga akan menghitung nilai avalance effect dari setiap algoritma. Pada akhirnya, kesimpulan akan ditarik dari kegiatan mengevaluasi performansi yang telah dilakukan.

III. HASIL DAN PEMBAHASAN

A. Skenario Eksperimen

Algoritma kriptografi yang dibandingkan dalam penelitian ini adalah algoritma Simon-Speck, Simeck, dan Skinny. Algoritma yang sudah berhasil disusun akan diunggah pada Raspberry Pi. Data yang digunakan untuk pengujian merupakan gabungan data yang diperoleh dari sensor-sensor perangkat IoT. Data hasil enkripsi yang menjadi pesan yang harus dikirim memiliki *output* yang berbeda-beda karena desain struktur dan operasi pada *round function* yang digunakan setiap algoritma juga berbeda-beda. Algoritma Simon-Speck, dan Simeck menggunakan struktur Feistel, sedangkan algoritma Skinny menggunakan struktur SPN. Pengambilan data dilakukan beberapa kali untuk mendapatkan data waktu enkripsi dan waktu dekripsi. Data yang digunakan untuk pengujian waktu enkripsi merupakan rata-rata waktu dari 30 data yang berhasil terenkripsi dan dikirim dari LoRa *Transmitter* ke Lora *Receiver*. *Ciphertext* yang dikirim selama komunikasi berlangsung berbentuk *hexadecimal* dan panjangnya bermacam-macam sesuai dengan panjang *plaintext* awal. Setelah *ciphertext* berhasil diterima oleh *Receiver*, *ciphertext* akan melalui proses dekripsi atau diubah menjadi *plaintext* atau pesan semula. Data yang digunakan untuk pengujian waktu dekripsi merupakan rata-rata waktu dari 30 data yang berhasil terkirim sampai ke Lora *Receiver*. Berikut adalah hasil dari input dan output setiap algoritma enkripsi. Berdasarkan Tabel 1, proses enkripsi pada keempat algoritma menggunakan ukuran kunci 128 bit dan ukuran blok 64 bit. Total panjang pesan (data sensor) yang harus dienkripsi sebesar 59 *byte*.

1) Algoritma Simon
   Pada algoritma Simon, jika panjang pesan dipotong setiap 20 *byte*, maka algoritma akan dijalankan untuk mengenkripsi 9 blok data sehingga jumlah *round* yang dijalankan sebanyak 396 putaran. Selanjutnya, panjang pesan yang dipotong setiap 30 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 352 putaran. Panjang pesan yang dipotong setiap 40 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 352 putaran.

2) Algoritma Speck
   Pada algoritma Speck, panjang pesan yang dipotong setiap 20 *byte* akan mengenkripsi 9 blok data sehingga jumlah *round* yang dijalankan sebanyak 243 putaran. Selanjutnya, panjang pesan yang dipotong setiap 30 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 216 putaran. Panjang pesan yang dipotong setiap 40 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 216 putaran.

3) Algoritma Simeck
   Pada pengujian algoritma Simeck, panjang pesan yang dipotong setiap 20 *byte* akan mengenkripsi 9 blok data sehingga jumlah *round* yang dijalankan sebanyak 396 putaran. Selanjutnya, panjang pesan yang dipotong setiap 30 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 352 putaran. Panjang pesan yang dipotong setiap 40 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 352 putaran.

4) Algoritma Skinny
   Terkait dengan pengujian pesan pada algoritma Skinny, panjang pesan yang dipotong setiap 20 *byte* akan mengenkripsi 9 blok data sehingga

jumlah *round* yang dijalankan sebanyak 324 putaran. Selanjutnya, panjang pesan yang dipotong setiap 30 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 288 putaran. Panjang pesan yang dipotong setiap 40 *byte* akan mengenkripsi 8 blok data sehingga jumlah *round* yang dijalankan sebanyak 288 putaran.

TABEL I. HASIL INPUT DAN OUTPUT PENELITIAN

Panjang Pesan (byte)	Plaintext	Ciphertext
Algoritma Simon		
20	rainfall0.0Humidity70.40Temperature31.8soil0.0velocity0.000	92810e1b494416abe849ef7fd6ffdaff1eac783018b3bcc9f6cb009bbbe491e77374d4bc6d6164d38df38d8f8539df47ea6482f49fd29c35335e5219a97a282a4430b043a5c3efdb
30	rainfall0.0Humidity69.80Temperature30.3soil0.0velocity0.000	92810e1b494416abe849ef7fd6ffdaff6e345591de98677d7ec5400446037cbb7e6245c20dfe8318be3d4bda3de3e2a0a1e2e245c7e16fe6eb4e3717100810e1
40	rainfall0.0Humidity71.20Temperature29.8soil0.0velocity0.000	92810e1b494416abe849ef7fd6ffdaff0d9bd2085543beb2504df9d0b72d28b0051b4f1bb6a01c05ea6482f49fd29c35335e5219a97a282a4430b043a5c3efdb
Algoritma Speck		
20	rainfall0.0Humidity70.40Temperature31.8soil0.0velocity0.000	d9332e972b8e94b7fcf3a26df38842580f9a527b1669654ec413b8b2d19dabd64b5f89745f1d7fe87c98d1fc6edfbe4ce9208a97bd253b6954cc357afd20b9c22eaea0314b5b5301
30	rainfall0.0Humidity69.80Temperature30.3soil0.0velocity0.000	d9332e972b8e94b7fcf3a26df38842587a2b25be078627c9d9243b1cffd88f6386bf6839b5bf0ec590fd701ac5b2d295a3c5c294f25464d6c286b5d561a1487a
40	rainfall0.0Humidity71.20Temperature29.8soil0.0velocity0.000	d9332e972b8e94b7fcf3a26df3884258d3357d6ae149693687b102abe0c612d50432cce577b0c2d3e9208a97bd253b6954cc357afd20b9c22eaea0314b5b5301
Algoritma Simeck		
20	rainfall0.0Humidity70.40Temperature31.8soil0.0velocity0.000	21525f41f8bb4ed7e549877b1f7c195f197dc2a7fd05a775ec2da179bd85b1599e95cf077cf22b855a4cce0ed6859912572f33d1c71288e3bd66d0323c811983b554078e5e15bf88
30	rainfall0.0Humidity69.80Temperature30.3soil0.0velocity0.000	21525f41f8bb4ed7e549877b1f7c195fac0d2b295ec4900b6ac90f9fc022e77358db4b4064c8a52e88d5bbd376315920c1b580cd2df1934e0c2f4b5f2870180e
40	rainfall0.0Humidity71.20Temperature29.8soil0.0velocity0.000	21525f41f8bb4ed7e549877b1f7c195f33d9283af82346b721391e137dfc2867fa40d769a3ec8c0a572f33d1c71288e3bd66d0323c811983b554078e5e15bf88
Algoritma Skinny		
20	rainfall0.0Humidity70.40Temperature31.8soil0.0velocity0.000	5c3fc556f26726ab9a44ac92b9c9fa5dea27bd97837c9a6ccf44eb863cbaa8481b2ac85dcf34f77638614eb1f0d26aeb704083928a8eebf18a91940b6d52e03550683f7897e28e98
30	rainfall0.0Humidity69.80Temperature30.3soil0.0velocity0.000	5c3fc556f26726ab9a44ac92b9c9fa5d5f4970725a52c88be59dceae6eec9a6d0c72a462d2aa54d58b71b62f014df457cd65c5f4b58cf3393a72292a035a42ef
40	rainfall0.0Humidity71.20Temperature29.8soil0.0velocity0.000	5c3fc556f26726ab9a44ac92b9c9fa5d4514f9e0b86f7ab3cb9da6ca5659111edfe6356e3ab2d134704083928a8eebf18a91940b6d52e03550683f7897e28e98

B. Pengujian Waktu Komputasi Proses Enkripsi

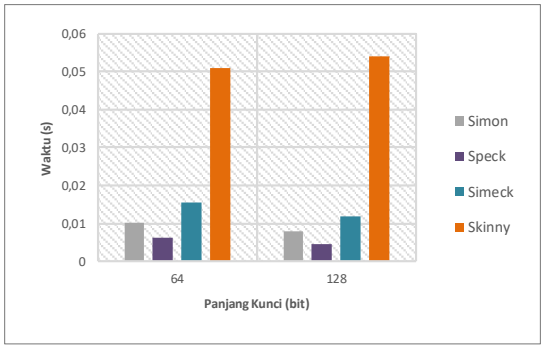
Waktu yang diperlukan perangkat untuk melakukan proses enkripsi merupakan waktu komputasi proses enkripsi. Lama waktu komputasi dipengaruhi oleh panjang pesan dan ukuran kunci yang digunakan oleh algoritma selama proses enkripsi dan dekripsi. Pengujian waktu enkripsi diamati berdasarkan panjang pesan yang berbeda-beda.

1) Panjang Pesan 20 *Byte*
   Gambar 3 menunjukkan bahwa jika menggunakan pesan yang panjangnya sebesar 20 *byte* dalam proses enkripsi, maka algoritma Skinny merupakan algoritma yang membutuhkan waktu paling lama pada ukuran kunci 64 dan 128 bit. Pada sisi sebaliknya, algoritma yang membutuhkan waktu paling sedikit atau paling cepat di antara keempat algoritma yaitu algoritma Speck. Pada ukuran kunci sebesar 64 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 87,5%. Selanjutnya, pada ukuran kunci sebesar 128 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 91,4%. Jadi, pada pengujian berdasarkan panjang pesan 20 byte, Algoritma Speck memiliki waktu enkripsi paling cepat, yaitu sebesar 0,00462585 detik ketika ukuran

kunci yang digunakan adalah 128 bit. Kemudian, Algoritma Skinny memiliki waktu paling lambat, yaitu selama 0,05393 detik pada saat ukuran kunci yang digunakan adalah 128 bit.

2) Panjang Pesan 40 *Byte*

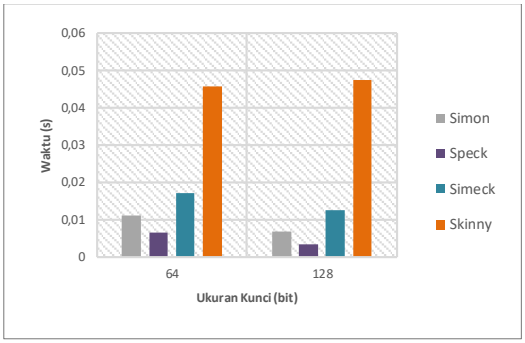
Grafik pada Gambar 4 menunjukkan bahwa jika kita menggunakan perpotongan panjang pesan sebesar 40 *byte* dalam proses enkripsi, maka algoritma Skinny merupakan algoritma yang membutuhkan waktu paling lama pada setiap ukuran kunci. Algoritma yang membutuhkan waktu paling sedikit atau paling cepat di antara algoritma lainnya yaitu algoritma Speck. Pada ukuran kunci sebesar 64 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 85,63%. Pada ukuran kunci sebesar 128 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 93,13%. Jadi, pada pengujian berdasarkan panjang pesan sebesar 40 byte, Algoritma Speck memiliki waktu enkripsi paling cepat, yaitu sebesar 0,00325 detik ketika ukuran kunci yang digunakan adalah 128 bit. Kemudian, algoritma Skinny menghabiskan waktu yang paling banyak atau paling lambat selama pengamatan, yaitu selama 0,0473 detik pada saat ukuran kunci yang digunakan adalah 128 bit.



Gambar 3. Waktu Enkripsi pada Panjang Pesan 20 Byte

C. *Pengujian Waktu Komputasi Proses Dekripsi*

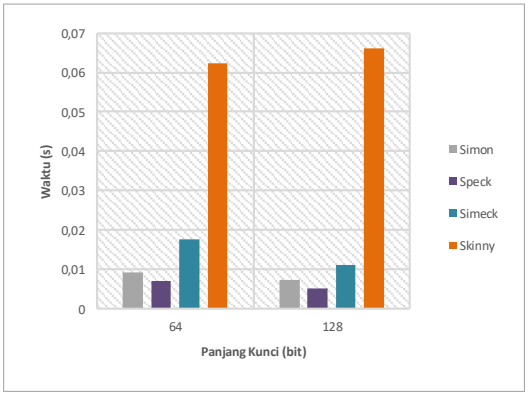
Sesuatu yang akan dianalisis selanjutnya adalah waktu komputasi yang dibutuhkan suatu algoritma untuk menyelesaikan satu kali proses dekripsi. Saat pengambilan data, kunci yang digunakan untuk proses dekripsi harus sama dengan kunci yang digunakan untuk proses enkripsi karena keempat algoritma termasuk ke dalam kriptografi simetris. Pengujian waktu dekripsi diamati berdasarkan panjang pesan/plaintext yang sama.



Gambar 4. Waktu Enkripsi pada Panjang Pesan 40 Byte

1) Panjang Pesan 20 Byte

Grafik pada Gambar 5 menunjukkan waktu yang dibutuhkan untuk proses dekripsi setiap algoritma pada pengamatan ciphertext dari plaintext yang panjangnya 20 byte. Grafik menunjukkan bahwa algoritma Skinny merupakan algoritma yang membutuhkan waktu paling lama untuk proses dekripsi pada ukuran kunci 64 dan 128 bit. Algoritma yang membutuhkan waktu paling sedikit atau paling cepat di setiap ukuran kunci adalah algoritma Speck. Pada ukuran kunci sebesar 64 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 88,36%. Lalu, pada ukuran kunci sebesar 128 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 92,14%. Oleh karena itu, pada pengujian berdasarkan skenario yang telah disebutkan sebelumnya, Algoritma Speck yang memiliki waktu dekripsi paling cepat, yaitu sebesar 0,00519 detik ketika ukuran kunci yang digunakan adalah 128 bit. Kemudian, selama pengamatan algoritma Skinny yang menghabiskan waktu yang paling banyak atau paling lambat, yaitu selama 0,066 detik pada saat ukuran kunci yang digunakan adalah 128 bit.

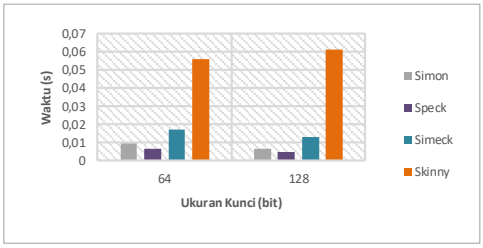


Gambar 5. Waktu Dekripsi pada Panjang Pesan 20 Byte

2) Panjang Pesan 40 *Byte*

Grafik pada Gambar 6 menunjukkan waktu yang dibutuhkan untuk proses dekripsi setiap algoritma pada pengamatan *ciphertext* dari

plaintext yang panjangnya 40 *byte*. Grafik menunjukkan bahwa algoritma Skinny merupakan algoritma yang membutuhkan waktu paling lama untuk proses dekripsi pada setiap ukuran kunci. Sedangkan, algoritma yang membutuhkan waktu paling sedikit atau paling cepat di setiap ukuran kunci merupakan algoritma Speck. Pada ukuran kunci sebesar 64 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 88,38%. Kemudian, pada ukuran kunci sebesar 128 bit, selisih waktu antara algoritma tercepat dan terlambat sebesar 92,35%. Oleh karena itu, pada pengujian berdasarkan skenario yang telah disebutkan sebelumnya, Algoritma Speck yang memiliki waktu dekripsi paling cepat, yaitu sebesar 0,0046 detik ketika ukuran kunci yang digunakan adalah 128 bit. Kemudian, algoritma Skinny yang menghabiskan waktu yang paling banyak atau paling lambat selama pengamatan, yaitu selama 0,06 detik pada saat ukuran kunci yang digunakan adalah 128 bit.

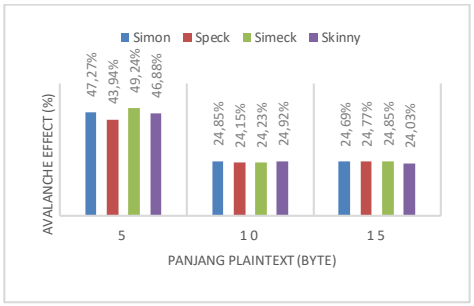


Gambar 6. Waktu Dekripsi pada Panjang Pesan 40 Byte

D. *Pengujian Avalanche Effect*

Terdapat dua metode untuk pengujian nilai avalanche effect pada penelitian ini, yaitu mengganti satu bit pada pesan awal dan mengganti 1 bit pada kunci awal.

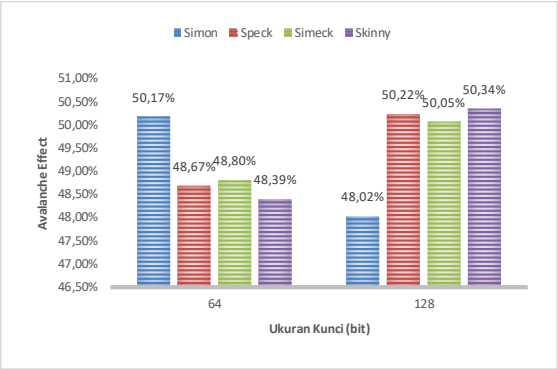
1) Pengujian Nilai *Avalanche Effect* dengan Perubahan 1 Bit pada *Plaintext*
Pengujian *avalance effect* kali ini dilakukan dengan cara melakukan enkripsi *plaintext* yang berbeda namun tetap menggunakan kunci yang sama. *Plaintext* asli akan diubah sebanyak satu bit dan dilakukan proses enkripsi. Proses enkripsi dilakukan menggunakan kunci yang sama. *Ciphertext* dengan *plaintext* asli akan dibandingkan dengan *ciphertext* dari hasil pengubahan plaintext sebanyak satu bit untuk memperoleh nilai *avalanche effect* Nilai *avalanche effect* yang ditampilkan pada grafik merupakan hasil rata-rata dari 30 kali pengambilan data. Gambar 7 merupakan grafik tingkat *avalanche effect* dari pengujian enkripsi pesan yang panjangnya 5, 10, 15 *byte* dan menggunakan *block size* 64 bit dan *key size* 128 bit. Pada gambar tersebut dapat terlihat bahwa algoritma Simeck memiliki tingkat keamanan paling tinggi dan Speck memiliki tingkat keamanan paling rendah. Nilai *avalanche effect* yang dimiliki algoritma Simeck dan Speck masing-masing sebesar 49,24% dan 43,94%, sehingga selisih diantara keduanya sebesar 5,3%. Pada panjang pesan sebesar 10 dan 15 *byte* memiliki rata-rata *avalanche effect* yang hampir sama, sebesar 24,53% dan 24,58%.



Gambar 7. Avalanche Effect Perubahan 1 Bit Plaintext

2) Pengujian Nilai *Avalanche Effect* dengan Perubahan 1 Bit pada *Plaintext*
Pengujian *avalance effect* dilakukan dengan cara mengenkripsi *plaintext* yang sama dan mengubah satu bit pada kunci enkripsinya. *Ciphertext* yang menggunakan kunci asli akan dibandingkan dengan *ciphertext* yang kunci enkripsinya telah diubah sebanyak satu bit untuk mendapatkan nilai *avalanche effect*. Nilai *avalanche effect* yang ditampilkan pada grafik merupakan hasil rata-rata dari 30 kali pengambilan data. Gambar 8 menunjukkan perbandingan nilai *avalanche effect* keempat algoritma. Algoritma Simon mengalami penurunan nilai a*valanche effect* sebesar 2,15% ketika ukuran kunci yang digunakan semakin besar. Algoritma Speck mengalami kenaikan nilai *avalanche effect* sebesar 1,55% ketika ukuran kunci diperbesar. Algoritma Simeck mengalami kenaikan nilai *avalanche effect* sebesar 1,26% ketika ukuran kunci diperbesar. Algoritma Skinny mengalami kenaikan nilai *avalanche effect* sebesar 1,95% ketika ukuran kunci diperbesar.

Algoritma Skinny memperoleh nilai *avalanche effect* terbaik ketika ukuran kunci yang digunakan sebesar 128 bit. Oleh karena itu, algoritma Skinny merupakan algoritma yang memiliki sistem keamanan paling baik di antara algoritma lainnya.



Gambar 8. *Avalanche Effect* Perubahan 1 Bit Kunci

IV. KESIMPULAN DAN SARAN

Pengujian algoritma Simon-Speck, Simeck, dan Skinny pada perangkat IoT berbasis LoRa berhasil dijalankan. Setiap algoritma memiliki hasil yang berbeda-beda yang disebabkan berbagai faktor. Perbedaan struktur desain dan operasi setiap algoritma membuat hasil enkripsi berbeda-beda. Lama proses enkripsi dan dekripsi dipengaruhi oleh banyaknya putaran (*round*) yang dijalan setiap algoritma pada ukuran blok dan panjang pesan yang berbeda-beda. Algoritma Speck memiliki waktu enkripsi dan dekripsi paling cepat, dan diikuti oleh algoritma Simon, Speck, dan Skinny.

Hasil dari pengujian *avalanche effect* menunjukkan bahwa kenaikan ukuran kunci tidak menjamin tingkat keamanan yang semakin tinggi. Tingkat keamanan *block cipher* bergantung kepada parameter dan struktur pada algoritma masing-masing. Selanjutnya, algoritma yang memiliki tingkat keamanan terbaik di antara keempat algoritma adalah algoritma Skinny yang menggunakan ukuran kunci 128 bit karena memiliki nilai *avalanche effect* yang paling tinggi. Berdasarkan hal tersebut dapat ditarik kesimpulan, jika pengguna ingin menghemat waktu dengan tingkat keamanan yang cukup, maka pengguna dapat memilih algoritma Speck untuk diimplementasikan. Akan tetapi, jika pengguna lebih mementingkan komunikasi data yang lebih aman dibandingkan waktu komputasi yang singkat, maka algoritma Skinny lebih cocok untuk diimplementasikan.

Saran untuk proses penelitian selanjutnya adalah menambahkan jenis algoritma *block cipher* lainnya untuk diimplementasikan dan dibandingkan agar diperoleh algoritma yang lebih efektif dan efisien untuk mengamankan komunikasi data antar perangkat IoT.

REFERENSI

[1] G. Lampropoulos, K. Siakas, and T. Anastasiadis, "Internet of Things in the Context of Industry 4.0: An Overview," *Int. J. Entrep. Knowl.*, vol. 7, no. 1, pp. 4–19, 2019, doi: 10.2478/ijek-2019-0001.

[2] Yudho Yudhanto and Abdul Azis, *Pengantar Teknologi Internet of Things (IoT)*, 1st ed. Surakarta: UNS Press, 2019.

[3] M. Dachyar, T. Y. M. Zagloel, and L. R. Saragih, "Knowledge growth and development: internet of things (IoT) research, 2006–2018," *Heliyon*, vol. 5, no. 8, p. e02264, 2019, doi: 10.1016/j.heliyon.2019.e02264.

[4] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on IoT security: Challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet of Things (Netherlands)*, vol. 11, p. 100227, 2020, doi: 10.1016/j.iot.2020.100227.

[5] Sathyan Munirathinam, "Chapter Six - Industry 4.0: Industrial Internet of Things (IIOT)," in *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, vol. 117, Pethuru Raj and P. Evangeline, Eds. Elsevier, 2020, pp. 129–164.

[6] A. Fotovvat, G. M. E. Rahman, S. S. Vedaei, and K. A. Wahid, "Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8279–8290, 2021, doi: 10.1109/JIOT.2020.3044526.

[7] D. Sehrawat and N. S. Gill, "Lightweight block ciphers for IoT based applications: a review," *Int. J. Appl. Eng. Res.*, vol. 13, no. 5, pp. 2258–2270, 2018.

[8] S. Singh Dhanda, B. Singh, P. Jindal, and S. S. Dhanda, "Lightweight Cryptography: A Solution to Secure IoT," vol. 112, pp. 1947–1980, 2020, doi: 10.1007/s11277-020-07134-3.

[9] S. M. S. Hussain, S. M. Farooq, and T. S. Ustun, "A Method for Achieving Confidentiality and Integrity in IEC 61850 GOOSE Messages; A Method for Achieving Confidentiality and Integrity in IEC 61850 GOOSE Messages," *IEEE Trans. Power Deliv.*, vol. 35, no. 5, 2020, doi: 10.1109/TPWRD.2020.2990760.

[10] S. Iyer, G. V. Bansod, V. Praveen Naidu, and S. Garg, "Implementation and Evaluation of Lightweight Ciphers in MQTT Environment," *3rd Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEECCOT 2018*, pp. 276–281, Dec. 2018, doi: 10.1109/ICEECCOT43722.2018.9001599.

[11] B. Y. Yustiarini, F. Dewanta, and H. H. Nuha, "A Comparative Method for Securing Internet of Things (IoT) Devices: AES vs Simon-Speck Encryptions," *2022 1st Int. Conf. Inf. Syst. Inf. Technol.*, pp. 392–396, Jul. 2022, doi: 10.1109/ICISIT54091.2022.9872666.

[12] T. Park, H. Seo, G. Lee, and H. Kim, "Efficient implementation of simeck family block cipher on 16-bit MSP430," *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 983–988, Jul. 2017, doi: 10.1109/ICUFN.2017.7993946.

[13] T. Ashur and A. Luykx, "An Account of the ISO/IEC Standardization of the Simon and Speck Block Cipher Families," *Secur. Ubiquitous Comput. Syst.*, pp. 63–78, 2021, doi: 10.1007/978-3-030-10591-4_4.

[14] J. Ge, Y. Xu, R. Liu, E. Si, N. Shang, and A. Wang, "Power attack and protected implementation on lightweight block cipher SKINNY," *Proc. - 13th Asia Jt. Conf. Inf. Secur. AsiaJCIS 2018*, pp. 69–74, Aug. 2018, doi: 10.1109/ASIAJCIS.2018.00020.

[15] L. Sliman, T. Omrani, Z. Tari, A. E. Samhat, and R. Rhouma, "Towards an ultra lightweight block ciphers for Internet of Things," *J. Inf. Secur. Appl.*, vol. 61, p. 102897, Sep. 2021, doi: 10.1016/J.JISA.2021.102897.