IMPLEMENTASI DAN ANALISIS *LIGHTWEIGHT CRYPTOGRAPHY* UNTUK *INTERNET OF THINGS* (IOT)

Fernando[1], Lukas[2]
[1,2] Cognitive Engineering Research Group (CERG),
Program Studi Teknik Elektro Fakultas Teknik
Universitas Katolik Indonesia Atma Jaya – Jakarta
e-mail: [1]ndodrum95@gmail.com, [2]lukas@atmajaya.ac.id

ABSTRAK

Pada era *Internet of Things* (IoT), keamanan data merupakan salah satu faktor penting. Data dapat diamankan dengan menggunakan algoritma kriptografi. Algoritma kriptografi tersebut mengubah pesan asli menjadi suatu kode rahasia (enkripsi) sedangkan proses sebaliknya disebut dekripsi. Data yang akan dienkripsi pada penelitian ini berupa *boot log file* yang terdapat pada *Raspberry Pi* dan *Intel Compute Stick*. *Boot log file* tersebut dienkripsi menggunakan algoritma *Data Encryption Standard* (DES) dan *Data Encryption Standard Lightweight* (DESL). Kedua algoritma tersebut memerlukan suatu kunci rahasia untuk melakukan enkripsi. Kunci rahasia yang digunakan untuk algoritma DES dan DESL lalu dienkripsi menggunakan algoritma *Rivest-Shamir-Adleman (RSA)*. Kemudian, data hasil enkripsi diunggah ke *cloud* dan dapat diunduh serta didekripsikan pada komputer *client* (*Virtual Machine*). Berdasarkan pengujian yang dilakukan pada penelitian ini, tidak diperoleh perbedaan waktu yang signifikan antara enkripsi menggunakan algoritma DES maupun DESL. Walaupun begitu, algoritma DESL lebih cocok digunakan pada perangkat IoT karena ukuran kode program yang lebih kecil dan prinsip *locality of reference* yang dimiliki karena hanya menggunakan satu buah *S-box* yang diulang sebanyak delapan kali setiap *round*-nya.

Kata kunci: DES, DESL, IoT, Kriptografi, *Raspberry Pi*, *Intel Compute Stick*.

ABSTRACT

Data security is an important factor in the era of Internet of Things (IoT). Data can be secured with the use of cryptography algorithm. That cryptography algorithm changes the original message into secret code (encryption), while the opposite process is called decryption. In this paper, boot log file in Raspberry Pi and Intel Compute Stick will be encrypted using Data Encryption Standard (DES) and Data Encryption Standard Lightweight (DESL) algorithm. Both algorithms need a secret key to do the encryption, then the secret key is encrypted using Rivest-Shamir-Adleman (RSA) algorithm. The encryption result will be uploaded to the cloud and can be downloaded and decrypted on the client computer (Virtual Machine). Based on the testing result, there are no significant differences in time of encryption between DES and DESL algorithm. Although there are no significant differences in time of encryption, DESL algorithm is preferable to use on IoT device because of its small code size and the locality of reference principle.

Keywords: *DES, DESL, IoT, Cryptography, Raspberry Pi, Intel Compute Stick.*

PENDAHULUAN

Perkembangan teknologi informasi yang dikenal dengan sebutan *Internet of Things* (IoT) telah membuat berjuta perangkat elektronik saling terhubung. Hal ini merupakan suatu keuntungan karena data yang terdapat pada suatu perangkat dapat dikirim ke perangkat lainnya secara *real time*. Data yang dikirimkan ada yang bersifat umum ataupun rahasia.

Data yang bersifat umum dapat dikirim begitu saja. Namun, data yang bersifat rahasia perlu dijaga keamanannya dan tidak boleh sampai diketahui oleh pihak luar. Terjaganya keamanan dan kerahasian data yang dikirimkan merupakan faktor penting pada era IoT. Salah satu solusi untuk menjamin keamanan dan kerahasian data adalah dengan menggunakan kriptografi.

Penelitian ini bertujuan untuk menganalisis algoritma kriptografi yang sesuai untuk perangkat IoT. Dipilihnya algoritma Data Encryption Standard (DES) dan Data Encryption Standard Lightweight (DESL) bukanlah suatu kebetulan. Algoritma DES yang dipublikasikan pada tahun 1977 dirancang untuk perangkat elektronik yang memiliki sumber daya terbatas (prosesor dan memori) sehigga memiliki kemiripan dengan sumber daya dari perangkat IoT. Pengembangan algoritma DES dengan menghilangkan dan mengganti beberapa tahap yang selanjutnya dikenal menjadi algoritma DESL ingin diuji perbedaannya bukan hanya dari segi memori yang digunakan, tetapi juga berapa besar selisih waktu yang digunakan untuk melakukan proses enkripsi dan dekripsi data..

TEORI DASAR

A. Kriptografi

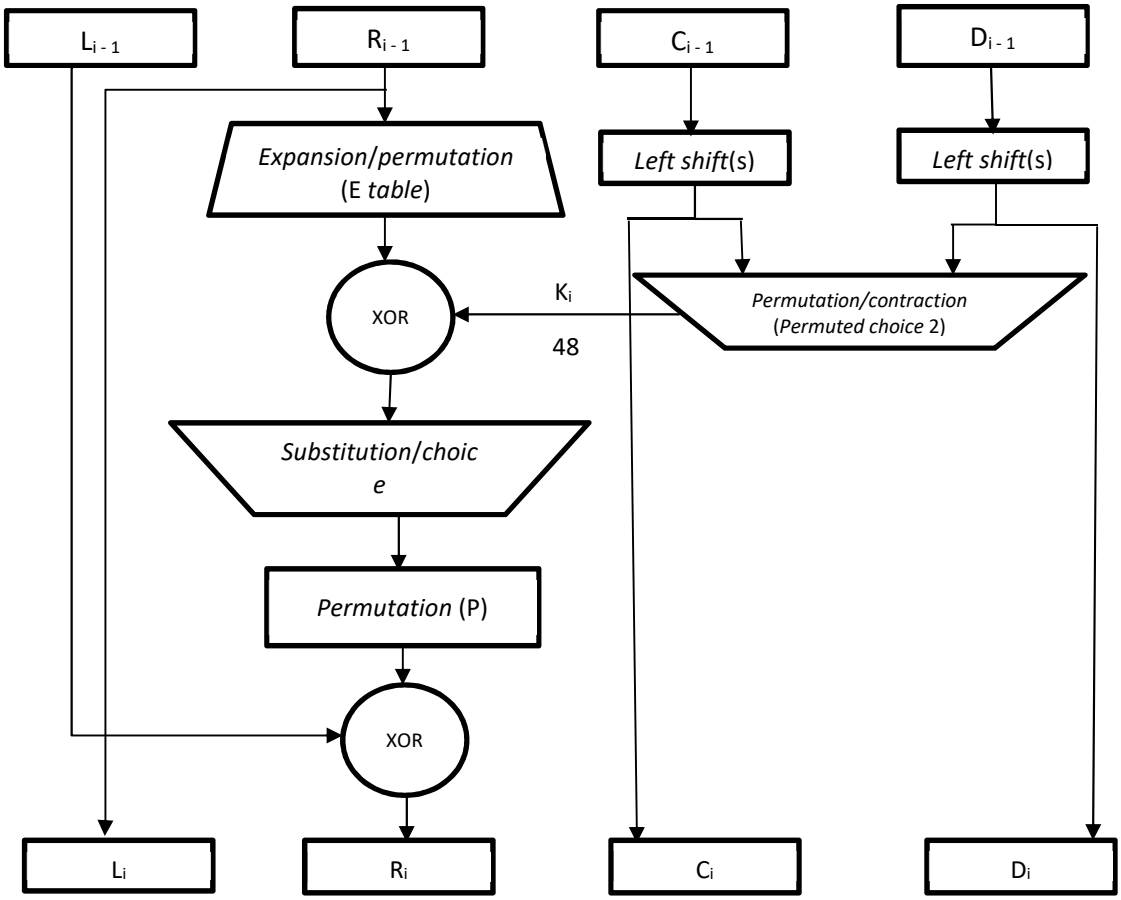
Kriptografi adalah ilmu ataupun seni yang mempelajari bagaimana membuat suatu pesan yang dikirim oleh pengirim dapat disampaikan kepada penerima dengan aman [4]. Pesan yang dikirim oleh pengirim perlu melalui suatu tahapan yang disebut enkripsi agar dapat diterima dengan aman. Enkripsi adalah proses untuk mengubah pesan asli (*plaintext*) menjadi pesan rahasia (*ciphertext*). Sebaliknya, proses untuk mengubah *ciphertext* menjadi *plaintext* disebut dekripsi.

Berdasarkan jenis kunci yang digunakan, sistem kriptografi dapat dibedakan menjadi *symmetric key* dan *asymmetric key*. Selain berdasarkan jenis kunci yang digunakan, sistem kriptografi juga dapat dibedakan menjadi *block cipher* dan *stream cipher* [5].

B. Algoritma *Data Encryption Standard*

Algoritma *Data Encryption Standard* (DES) mengubah *plaintext* menjadi *ciphertext* per blok data (*block cipher*) yang besarnya sama yaitu 64 *bit* dan menggunakan kunci yang besarnya 56 *bit*. Sebenarnya, kunci yang digunakan adalah 64 *bit* dengan 8 *bit* lainnya digunakan sebagai *parity bit*. Secara keseluruhan, tahapan pada algoritma DES dapat dibagi menjadi tiga bagian utama yaitu *initial permutation*, *round function*, dan *inverse initial permutation*.

Round function berlangsung sebanyak 16 kali. Masukan untuk setiap *round* merupakan dua buah blok data yang ukurannya masing-masing 32 *bit* yaitu L_{i-1} dan R_{i-1}. Masukan R kemudian diacak susunannya dan diperbesar ukurannya menjadi 48 *bit*. Setelah masukan R menjadi 48 *bit*, kemudian R di-XOR-kan dengan kunci. pada tiap *round* yang juga sebesar 48 *bit*.



Gambar 1 Struktur per round algoritma DES [6]

Kunci yang digunakan pada setiap *round* didapat dari 56 *bit* kunci yang dibagi menjadi dua bagian (C_{i-1} dan D_{i-1}) sebesar 28 *bit* setiap bagiannya. Untuk setiap *round*, masing-masing bagian (C_{i-1} dan D_{i-1}) dari kunci dilakukan *circular shift left* dan kemudian diperbesar ukurannya menjadi 48 *bit*. Struktur algoritma DES dapat dilihat pada Gambar 1.

C. Algoritma *Data Encryption Standard Lightweight*

Algoritma *Data Encryption Standard Lightweight* (DESL) adalah algoritma kriptografi yang dikembangkan dari algoritma DES. Pada dasarnya algoritma DESL sama dengan algoritma DES, perbedaannya terletak pada *S-box* yang digunakan. Selain itu, perbedaan antara DES dengan DESL terletak pada proses *initial permutation* dan *inverse initial permutation* yang dihilangkan pada algoritma DESL [2].

DES menggunakan delapan buah *S-box* yang berbeda pada setiap *round* sedangkan DESL hanya menggunakan sebuah *S-box* yang diulang sebanyak delapan kali setiap *round*. Penggunaan *S-box* yang hanya terdiri dari satu jenis pada DESL membuat *memory footprint* program DESL lebih kecil dibandingkan dengan DES.

D. Algoritma *Rivest-Shamir-Adleman*

Algoritma *Rivest-Shamir-Adleman* (RSA) adalah algoritma kriptografi *asymmetric key*. Dikembangkan tahun 1977 di MIT dan dipublikasikan tahun 1978 oleh Ron Rivest, Adi Shamir, dan Len Adleman. algoritma RSA terdiri dari tiga tahapan yaitu pembentukan kunci (*key generation),* enkripsi dan dekripsi yang ditampilkan pada Gambar 2.

	KEY GENERATION	
Pilih p, q		p dan q bilangan prima, p ≠ q
Hitung n = p x q		
Hitung $\phi(n) = (p-1) \times (q-1)$		
Pilih sebuah bilangan bulat positif e		$\text{FPB}(\phi(n), e) = 1; 1 < e < \phi(n)$
Hitung d		$d \equiv e^{-1} \bmod \phi(n)$
Public key		PU = {e, n}
Private key		PR = {d, n}
	ENKRIPSI	
Plaintext		$M < n$
Ciphertext		$C = M^e \bmod n$
	DEKRIPSI	
Ciphertext		C
Plaintext		$M = C^d \bmod n$

Gambar 2 Algoritma RSA [6]

E. Internet of Things

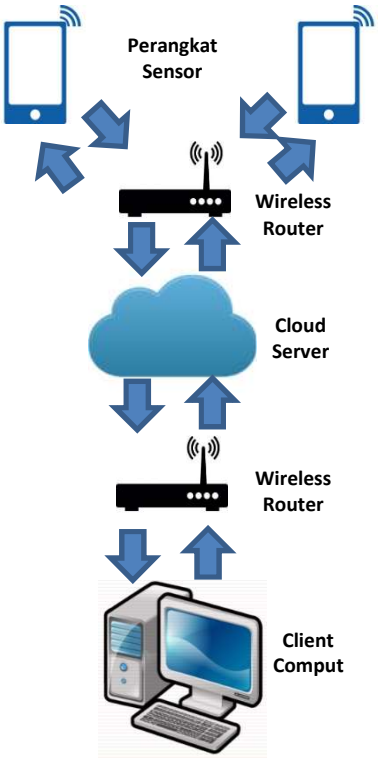
Istilah *Internet of Things* (IoT) adalah sebuah konsep dimana suatu objek memiliki kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan interaksi manusia ke manusia atau manusia ke komputer. Konsep ini akan meningkatkan kapasitas data yang dihasilkan dan diolah pada setiap perangkat elektronik 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 dapat disimpan pada penyimpanan internal.

PERANCANGAN SISTEM

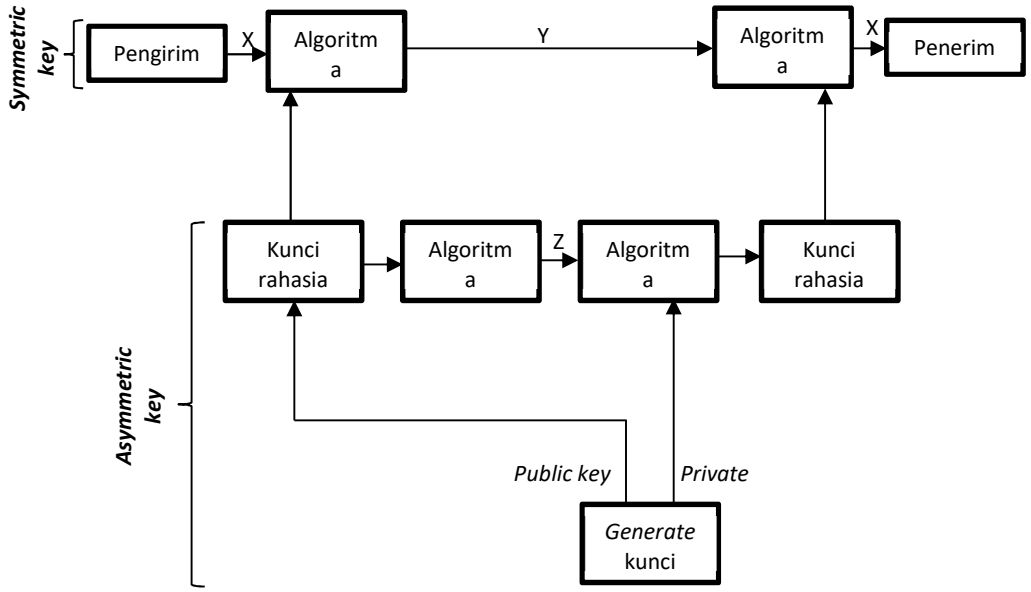
A. Cara Kerja

Raspberry Pi dan *Intel Compute Stick* yang digunakan pada penelitian ini merupakan sebuah mikrokomputer yang difungsikan sebagai sebuah sensor yang



Gambar 3 Topologi konektivitas perangkat

dapat mengirim data secara berkala ke *server*. Spesifikasi dari *Raspberry Pi* dan *Intel Compute Stick* yang digunakan tidak terlalu tinggi, sehingga cocok untuk digunakan sebagai perangkat sensor yang terkoneksi pada era IoT. Gambar 3 menunjukkan topologi konektivitas perangkat pada penelitian.



Gambar 4 Proses enkripsi dan dekripsi

Setiap kali *Raspberry Pi* atau *Intel Compute Stick* dinyalakan, program akan mengenkripsi data berupa *boot log file* yang menunjukkan catatan aktivitas saat sistem operasi mulai dijalankan. Program enkripsi yang dijalankan terdiri atas dua buah program, yaitu program pertama dengan menggunakan algoritma DES dan program kedua dengan menggunakan algoritma DESL. Masing-masing program akan membentuk *file* baru berupa:

1. *File* yang sudah terenkripsi (*ciphertext*) dari *plaintext* berupa *boot log file*.
2. *File* yang berisi waktu yang digunakan selama proses enkripsi.

Algoritma DES dan DESL memerlukan suatu kunci rahasia untuk melakukan enkripsi. Kunci rahasia dibentuk secara otomatis menggunakan 4 karakter pertama digabung dengan 4 karakter terakhir dari data yang terdapat pada *boot log file*. Kunci rahasia perlu terjaga kerahasiaannya, sehingga kunci rahasia akan dienkripsi menggunakan *public key* dengan algoritma RSA.

Public key yang digunakan diunduh dari *cloud* yaitu AWS. Sebelumnya, *public key* sudah terlebih dahulu di-*generate* pada sisi penerima bersama dengan *private key*. Namun, hanya *public key* yang diunggah ke AWS.

Setelah proses enkripsi selesai dan jika *Raspberry Pi* atau *Intel Compute Stick* terhubung dengan internet, *file* yang sudah terenkripsi (*boot log file* dan kunci rahasia) diunggah ke AWS. *File* yang sudah terdapat pada AWS, dapat diunduh dan didekripsikan menggunakan program dekripsi yang terdapat pada *virtual machine*.

Program dekripsi memerlukan suatu masukan berupa kunci rahasia yang harus didekripsi terlebih dahulu menggunakan *private key* dengan algoritma RSA. Kemudian kunci rahasia yang sudah didekripsi digunakan untuk mendekripsi *ciphertext* dari *boot log file*. Setelah proses dekripsi selesai, kunci rahasia yang telah diunduh akan terhapus secara otomatis agar kerahasian kunci tetap terjaga. Gambar 4 menunjukkan proses enkripsi dan dekripsi pada penelitian secara ringkas.

B. Pembuatan *Bucket*

Salah satu fitur yang terdapat pada AWS adalah *Amazon Simple Storage Service* (AWS S3). Agar dapat menggunakan layanan AWS S3, pengguna perlu membuat akun terlebih dahulu ke AWS.

Untuk menyimpan data pada AWS S3, terlebih dahulu harus dibuat *bucket* sebagai tempat penyimpanan. Nama *bucket* harus unik, artinya tidak boleh sama dengan nama

```
@reboot sleep 10; python3 ~/Programs/bootMonitor.py
@reboot sleep 20; python3 ~/Programs/encryptAutorun.py
```

Gambar 5 Sintaks untuk menjalankan program saat start up linux [3]

bucket yang sudah ada. Langkah-langkah untuk membuat *bucket* pada AWS S3 adalah sebagai berikut:

1. Pada *browser*, ketik alamat *https://aws.amazon.com/s3/* kemudi-an klik "*Get started with Amazon S3*" pada alamat yang muncul. Pada halaman selanjutnya, pilih "*I am a returning user and my password is*", masukkan *email* dan *password*.
2. Klik *create bucket* pada halaman yang muncul setelah *login*.
3. Isi *bucket name* dengan nama yang *bucket* yang ingin dibuat, pilih *Region* menjadi "*Asia Pasific (Singapore)*" lalu klik *create*.

Lakukan hal yang sama untuk membuat *bucket* lainnya. Pada penelitian ini akan terdapat empat buah *bucket* yang akan dibuat, yaitu "`ta1e`", "`ta2c`", "`kodeprogramlinux`", dan "`kodeprogramwindows`". *Bucket* "`ta1e`" dan "`ta2c`" digunakan untuk menampung data hasil enkripsi dan *public key* yang digunakan. Bucket "`kodeprogramwindows`", dan "`kodeprogramlinux`" digunakan untuk menampung kode program yang digunakan untuk mengenkripsi atau mendekripsikan data.

C. Pembuatan *Modul*

Untuk menjalankan sistem kriptografi pada penelitian ini, perlu dibuat beberapa modul (kode program) menggunakan bahasa *Python*. Tidak semua modul sama antara penerima atau pengirim. Modul yang telah dibuat kemudian diunggah ke AWS S3. Oleh karena itu, perlu dibuat suatu modul tambahan berupa modul `setup.py` yang digunakan untuk mengunduh kode program secara otomatis dari AWS S3 baik pada sisi pengirim ataupun penerima.

Modul yang terdapat pada kedua pihak antara lain `AWS.py`, `connection.py`, `converter.py`, `DES.py`, `DESL.py`, `directory.py`, `iofile.py`, `log.py`, `padding.py`, `prime.py`, `RSA.py`, `vigenare.py`, dan `selectMode.py`. Modul yang terdapat hanya pada pihak pengirim yaitu `bootMonitor.py` dan `encryptAutorun.py` sedangkan modul yang terdapat pada penerima adalah `keyAutorun.py` dan `decryptAutorun.py`.

Modul `DES.py`, `DESL.py`, `RSA.py`, dan `vigenere.py` merupakan modul yang berisi tentang algoritma kriptografi baik enkripsi maupun dekripsi. Modul `connection.py` berisi fungsi untuk menguji perangkat terhubung internet atau tidak. Modul `converter.py` berisi fungsi untuk melakukan konversi data seperti konversi dari desimal ke biner atau sebaliknya. Modul `iofile.py` berisi fungsi untuk melakukan penulisan atau pembacaan suatu data dari atau ke *file* tertentu. Modul `log.py` berisikan catatan mengenai proses enkripsi dan dekripsi yang dijalankan. Modul `AWS.py` berisi fungsi yang berhubungan dengan AWS S3 seperti menggungah atau mengunduh data dari atau ke AWS S3. Modul `prime.py` berfungsi untuk menguji apakah suatu bilangan merupakan bilangan prima. Modul `padding.py` berfungsi untuk melakukan penambahan jumlah *bit* data hingga ukuran tertentu.

D. Eksekusi Kode *Python* Otomatis saat *Startup* Perangkat

Setelah modul `setup.py` dijalankan dan semua kode program telah diunduh, terdapat beberapa modul yang harus dijalankan secara otomatis saat perangkat dinyalakan. Modul tersebut yaitu `bootMonitor.py` dan

```
TIMEOUT /T 1
START "" "C:\Users\ndodr\Programs\bootMonitor.py
"TIMEOUT /T 15
START "" "C:\Users\ndodr\Programs\encryptAutorun.py"
```

Gambar 6 Sintaks pada StartupOrder.bat

`encryptAutorun.py`. Terdapat perbedaan cara untuk membuat kode program dijalankan secara otomatis saat perangkat dinyalakan antara sistem operasi berbasis *Windows* dan sistem operasi berbasis *Linux*.

Pada sistem operasi berbasis *Linux*, agar kode program dapat dijalankan secara otomatis saat perangkat dinyalakan, maka dapat menggunakan *command* `crontab -e` pada *terminal*. Tekan *Enter*, kemudian tambahkan sintaks seperti pada Gambar 5 pada akhir dari `crontab -e` dan simpan setelahnya.

Kode `@reboot` menyatakan bahwa program akan dieksekusi setiap perangkat dinyalakan. Kode `sleep 10` dan `sleep 20` berfungsi untuk mem-berikan waktu jeda 10 detik dan 20 detik setelah sistem operasi beroperasi penuh. Waktu jeda ini perlu diberikan, karena program yang akan dieksekusi memerlukan masukan berupa data pada perangkat setelah sistem operasi berjalan penuh. Kode `python3` diikuti dengan *path* dimana kode program berada menyatakan bahwa program yang akan dieksekusi menggunakan bahasa pemrograman *Python*. Perlu diperhatikan karena modul `enryptAutorun.py` memerlukan data dari modul `bootMonitor.py`, maka modul `bootMonitor.py` perlu dijalankan terlebih dahulu.

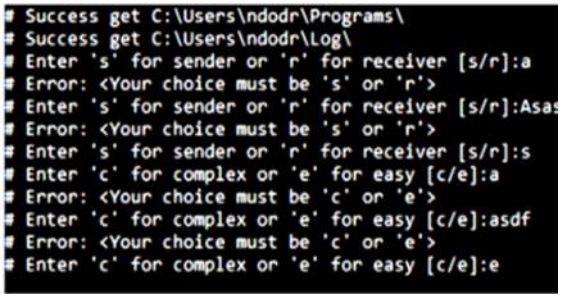
Langkah – langkah tersebut dapat dilakukan dengan menggunakan bahasa pemrograman *Python* dengan menggunakan modul *crontab*. Modul *crontab* perlu diunduh terlebih dahulu dengan mengetikkan `pip3 install python-crontab` pada *terminal* atau *command promt*. Pembuatan kode *python* untuk dieksekusi secara otomatis saat perangkat dinyalakan ditambahkan ke dalam modul `setup.py`.

Pada sistem operasi *Windows*, kode program dapat dieksekusi secara otomatis saat perangkat dinyalakan dengan menempatkan kode program tersebut pada *directory* `~\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Start-up`. Kode program yang terdapat pada *directory* tersebut akan dijalankan secara bersamaan setelah perangkat dinyalakan. Agar kode program dapat dijalankan secara berurutan antara kode yang satu dengan kode yang lainnya, perlu dibuat suatu *file* dengan nama `StartupOrder.bat` pada *directory* tersebut yang didalamnya terdapat sintaks seperti pada Gambar 6.

PENGUJIAN SISTEM

A. Pengujian `setup.py`

Jika modul `setup.py` dijalankan, modul tersebut akan meminta masukan apakah perangkat akan digunakan sebagai pengirim atau penerima serta mode yang akan digunakan pada sistem. Apabila pengguna memasukkan pilihan yang lain selain yang diminta, maka program akan menampilkan pesan peringatan dan melakukan pengulangan untuk meminta pengguna memasukkan masukan yang sesuai. Pengulangan tersebut dapat dilihat pada Gambar 7.



Gambar 7. Pengujian masukan yang diberikan pada modul setup.py

Program akan berhenti jika perangkat tidak tersambung dengan internet. Jika perangkat terhubung dengan internet, modul yang diperlukan akan diunduh dari AWS S3 sesuai dengan sistem operasi yang dijalankan dan mode yang dipilih.

B. Pengujian Data Hasil Enkripsi dan Dekripsi

Boot log file hasil pengujian pada *Raspberry Pi* dengan sistem operasi *Rasbian* dapat dilihat pada Gambar 8 sedangkan hasil pengujian *boot log file* pada *Intel Compute Stick* dengan sistem operasi *Windows* pada Gambar 9. *Boot log file* tersebut kemudian dienkripsi menggunakan algoritma DES dan DESL yang kemudian diunggah ke AWS S3.

```
15:49:05 Saturday 06/17/17
Ip address: 192.168.0.115
Startup finished in 1.766s
(kernel) + 8.401 (userspace)
= 10.168s
```

Gambar 8 Boot log file pada Raspberry Pi

Data hasil enkripsi diunduh dan kemudian didekripsi pada komputer penerima, berupa *Virtual Machine*. Didapat berdasarkan hasil pengujian, data sebelum enkripsi sama dengan data sesudah dekripsi.

Gambar 9 Boot log file pada Intel Compute Stick

C. Pengujian Waktu Eksekusi Program

Pada proses enkripsi maupun dekripsi dilakukan pengujian waktu yang digunakan. Waktu yang digunakan antara proses enkripsi maupun dekripsi dengan besar data yang sama terdapat perbedaan. Tabel 1 menunjukkan hasil pengujian waktu pada *Raspberry Pi* sebagai pengirim (enkripsi) sedangkan Tabel 2 pada *Intel Compute Stick* sebagai pengirim dengan *Virtual Machine* sebagai penerima (dekripsi).

Besar data berupa banyaknya karakter pada *boot log file* di *Raspberry Pi* adalah 115 karakter sedangkan pada *Intel Compute Stick* adalah 3620 karakter. Dari data pada Tabel 1 dan Tabel 2, dapat dihitung nilai rata-rata waktu yang digunakan dari setiap algoritma baik untuk enkripsi maupun dekripsi. Nilai rata-rata waktu hasil pengukuran dapat dilihat pada Tabel 3.

SIMPULAN

Berdasarkan hasil penelitian dan pengujian, didapat bahwa waktu yang diperlukan untuk melakukan proses enkripsi atau dekripsi menggunakan algoritma DESL tidak terdapat perbedaan yang signifikan dengan algoritma DES. Perbedaan waktu yang diperoleh untuk melakukan enkripsi dengan jumlah data 115 karakter adalah 2.4ms (1.18%) sedangkan untuk jumlah data 3620 karakter adalah 2ms (0.2%). Walaupun begitu, algoritma DESL lebih cocok

digunakan pada perangkat IoT karena ukuran kode program yang lebih kecil dan prinsip *locality of reference* yang dimiliki karena hanya menggunakan satu buah *S-box* yang diulang sebanyak delapan kali setiap *round*-nya.

Tabel 1 Pengujian waktu dengan *Raspberry Pi* sebagai pengirim

Pengujian Ke	Waktu yang digunakan			
	Algoritma DES		Algoritma DESL	
	Enkripsi (s)	Dekripsi (s)	Enkripsi (s)	Dekripsi (s)
1	0.2033	0.0320	0.2004	0.0269
2	0.2046	0.0344	0.2017	0.0248
3	0.2011	0.0317	0.1992	0.0245
4	0.2018	0.0297	0.1998	0.0247
5	0.2018	0.0322	0.2027	0.0257
6	0.2054	0.0324	0.2027	0.0243
7	0.2026	0.0323	0.1990	0.0245
8	0.2038	0.0386	0.2003	0.0340
9	0.2033	0.0541	0.2013	0.0286
10	0.2031	0.0513	0.1998	0.0365

Tabel 2 Pengujian waktu dengan *Intel Compute Stick* sebagai pengirim

Pengujian Ke	Waktu yang digunakan			
	Algoritma DES		Algoritma DESL	
	Enkripsi (s)	Dekripsi (s)	Enkripsi (s)	Dekripsi (s)
1	0.9636	0.7204	0.9296	0.6991
2	0.9596	0.8826	0.9276	0.8446
3	0.9366	0.7346	0.9476	0.7148
4	0.9366	0.7312	0.9252	0.7107
5	0.9406	0.7157	0.9286	0.7287
6	0.9386	0.7224	0.9396	0.7008
7	0.9826	0.7210	0.9546	0.7055
8	0.9556	0.7656	1.0297	0.9263
9	0.9616	0.7205	1.0097	0.7112
10	0.9536	0.7252	0.9566	0.7217

Tabel 3 Nilai rata-rata waktu yang digunakan

Pengujian Ke	Waktu rata-rata				Jumlah Karakter
	Algoritma DES		Algoritma DESL		
	Enkripsi(s)	Dekripsi(s)	Enkripsi(s)	Dekripsi(s)	
Raspberry Pi	0.2031	0.0369	0.2007	0.0284	115
Intel Compute Stick	0.9529	0.7439	0.9549	0.7464	3620

DAFTAR PUSTAKA

[1] Munir, R. 2014. *Matematika Diskrit Revisi Kelima*. Bandung: Penerbit Informatika.

[2] Poschmann, A. Y. 2009. *Lightweight Cryptography: Cryptographic Engineering for a Pervasive World*. Disertasi. Bochum: Faculty of Electrical Engineering and Information Technology Ruhr.

[3] Ray, D. S. dan Ray, D. S. 2015. *Visual Quickstart Guide Unix and Linux*, Fifth Edition. Utah: Peachpit Press.

[4] Scheneir, B. 1996. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Second Edition. New York: John Wiley & Sons, Inc.

[5] Setyaningsih, E. 2015. *Kriptografi dan Implementasinya Menggunakan Matlab*. Yogyakarta: Penerbit Andi.

[6] Stallings, W. 2011. *Cryptography and Network Security,* Fifth Edition. New Jersey: Pearson Education, Inc.