Two Step Authentication dengan RFID dan algortima *Time-Based One Time Password* pada Smart Lock

Hatami Ra'is Bukhari[1], Vera Suryani[2], Erwid Musthofa Jadied[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[4]Divisi Digital Service PT Telekomunikasi Indonesia
[1]hatamirais@students.telkomuniversity.ac.id, [2]verasuryani@telkomuniversity.ac.id,
[3]jadied@telkomuniversity.ac.id

Abstrak

Kebanyakan perangkat *smart lock* menyediakan fitur dimana pengguna dapat mengatur konfigurasi *smart lock* mulai dari membuat kunci, membuka kunci, mendaftarkan pengguna baru melalui jaringan seperti melalui internet dan Bluetooth. Hal ini tentunya dapat memudahkan *cyber criminal* untuk mencari celah ke *smart lock* dan mengambil semua data yang ada. Pada tugas akhir ini, penulis merancang perangkat *smart lock* yang dapat bekerja dengan sendirinya tanpa koneksi ke jaringan lainnya dengan menggunakan Raspberry Pi 3 B, RFID dan Time Based One-Time (TOTP). Semua proses mulai dari mendaftarkan kartu, melakukan autentikasi RFID dan autentikasi TOTP, mengendalikan kunci dan komponen lainnya, dan menyimpan data di lakukan dari Raspberry Pi. Dengan menggunakan 5 pengujian yaitu uji fungsional, akurasi deteksi, *code response*, keamanan, dan *overhead* penulis mendapatkan hasil berupa perangakat yang dapat beroperasi secara mandiri yang hanya membutuhkan waktu selama 3 detik untuk membuka kunci walaupun memerlukan 2 autentikasi.

Kata kunci: smart lock, raspberry pi 3 b+ , rfid, TOTP

Abstract

Most smart lock devices provide features where users can set the smart lock configuration starting from creating a lock, unlocking it, registering new users via networks such as via the internet and Bluetooth. This certainly makes it easier for cyber criminals to find vulnerability in the smart lock and retrieve all existing data. In this final project, the authors designed a smart lock device that can work independently without connection to other networks using the Raspberry Pi 3 B, RFID and Time Based One-Time (TOTP). All processes from registering the card, performing RFID and TOTP authentication, controlling keys and other components, and storing data are carried out from the Raspberry Pi. By using 5 tests, namely functional tests, detection accuracy, code response, security, and overhead, the writer gets results in the form of a device that can operate independently which only takes 3 seconds to open the lock even though it requires 2 authentications.

Keyword: smart lock, raspberry pi 3 b+, rfid, TOTP

1. Pendahuluan

Di era teknologi yang semakin berkembang ini, penggunaan teknologi IoT di bidang *smart home* khususnya pada *smart lock* semakin bertambah banyak. Dengan mengutip data yang di dapat dari "Smart Lock Market Size, Share, Industry Research Report 2027" oleh Grand View Research ada sekitar 7 juta perangkat *smart lock* yang terjual di tahun 2019 [1]. Perangkat smart lock umumnya terhubung dengan bluetooth atau wifi agar bisa di awasi dan kendalikan dengan *smartphone* dan terhubung dengan *virtual assistance* AI seperti Amazon Alexa dan Google Assistant. Metode autentikasi yang popular di gunakan adalah dengan password baik dengan keypad atau touchscreen dan juga dengan fingerprint [2].

Pada umumnya, semakin canggih perangkat elektronik semakin rentan juga keamanannya. Hal ini juga terjadi di smart lock. Pada 11 desember 2019 tim dari F-Secure Labs menemukan celah di KeyWe smart lock dengan mengeksploitasi Bluetooth yang digunakan perangakat KeyWe untuk mengautentikasi user yang terdaftar di dalam sistem. Ekspolit yang di temukan adalah dengan *intercept* pesan yang akan dikirim dari sistem ke user melalui salah satu class yang ada di dalam program [3].

Latar Belakang

Tema tugas akhir ini adalah IoT *security* pada *smart home device* untuk *smart lock*. Konsep penggunaan RFID untuk smart lock sebenarnya sudah lama di terapkan dan sudah banyak di gunakan seperti di hotel – hotel. Namun banyak perangkat *smart lock* ini hanya menggunakan *passcode*, media NFC, atau media RFID untuk mengautentikasi user, karena ini penulis ingin meneliti pengaruh apa jika algoritma TOTP di gunakan sebagai *two step authentication* pada *smart lock* yang menggunakan RFID dengan memperhatikan masalah fungsionalitas,

akurasi deteksi kode otp, *code response* dan *overhead* di sistem yang berjalan dengan sendirinya tanpa koneksi dengan perangkat lainnya. Perangkat *smart lock* pada tugas akhir ini di rancang untuk menyelesaikan masalah pada *smart lock* yang dapat di tembus oleh hacker dengan mengeksploit komunikasi antara s*mart lock* dengan pengguna melalui perantara wireless. Penulis memilih RFID langkah awal autentikasi karena penggunaan arus tenaga yang rendah, dan algortima TOTP di pilih karena kemampuannya untuk membuat *passcode* baru yang hanya berlaku selama 30 detik. Detail untuk pemillihan RFID dan algoritma TOTP di bahas pada studi literatur.

Topik dan Batasannya

Topik pada tugas akhir ini adalah IoT *security* pada *smart home device*, yaitu *smart lock*. Perangkat *smart lock* yang di rangkai pada tugas akhir ini adalah berupa perangkat *smart lock* dengan menggunakan RFID dan TOTP sebagai 2 proses autentikasi pengguna untuk membuka kunci, semua proses mulai dari *user registration, user authentication, data storage,* dan *sensor controller* akan di tangani oleh *single-board computer* berupa Raspberry Pi 3 model B+ yang tidak terkoneksi dengan jaringan manapun. Semua sensor akan di kendalikan oleh Raspberry Pi 3 model B+ melalui kabel jumper dan admin bisa mengakses Raspberry Pi secara langsung dengan menggunakan monitor dan perangkat peripheral lainnya atau dengan menggunakan *remote access* [4].

Tugas akhir ini memfokuskan ke dalam penelitian tentang bagaimana *Time-Based One Time Password* (TOTP) akan berpengaruh ke dalam *Smart Lock* yang bekerja dengan sendirinya tanpa memerlukan koneksi dengan perangkat atau jaringan lainnya. Hasil akhir dari TA ini yang berupa rangkaian perangkat IoT di bidang *smart lock* ini, belum bisa di gunakan untuk pemakaian *smart lock* di kehidupan keseharian karena kurangnya fitur yang mengutamakan kenyamanan *end user* seperti tidak *compact*, tidak stabil, dan lebih cocok sebagai tahap awal pengembangan perangkat (*prototype*).

Hal yang menyebabkan akhir penelitian TA ini tidak bisa di terapkan ke kehidupan sehari-hari seperti yang ada di atas adalah karena kurangnya sumber daya baik dari pengalaman penulis dalam rangkaian elektronik dan pemograman dan juga segi keuangan yang di perlukan agar hasil akhir dari TA ini bisa langsung di gunakan di kehidupan sehari-hari.

Tujuan

Dengan mengangangkat masalah utama pada tugas akhir ini, yaitu celah keamanan yang ada di komunikasi antara perangkat *smart lock* dan pengguna. Tujuan dari tugas akhir ini adalah merancang perangkat *smart lock* dengan menggunakan RFID dan TOTP yang dapat berkerja dengan sendirinya tanpa koneksi dengan perangkat atau jaringan lainnya untuk proses autentikasi RFID dan kode OTP dan menyimpan data RFID dan *secret* untuk setiap RFID yang terdaftar. Penulis melakukan pengujian dan analisa dengan 5 parameter:

- Uji fungsional: setiap hardware yang terhubung dengan perangkat dapat berkerja satu sama lain
- Akurasi deteksi: seberapa akurat kode yang di dapat dari TOTP adalah kode yang benar dengan menggunakan 2 aplikasi autentikasi yang berbeda platform
- Code response: seberapa lama program memproses semua fungsi dari awal kartu RFID terdeteksi hingga membuka kunci.
- Keamanan: perangkat bisa menjaga keamanan smart lock dari serangan yang umum di gunakan baik untuk RFID dan TOTP
- Overhead: mengukur sebarapa banyak program menggunakan sumber daya CPU dan RAM di sistem

2. Studi Terkait

Tabel 1 Studi terkait

Nama penelitian	Fokus penelitian	Penulis
Smart Locks: Lessons for Securing Commodity Internet of Things Devices	Masalah keamanan yang di temukan perangkat smart lock yang umum di gunakan.	Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, David Wagner
Smart Digital Door Lock System Using Bluetooth Technology	Smart lock yang menggunakan Bluetooth dan cloud computing	Siddhi Kavde, Riddhi Kavde, Sonali Bodare, Gauri Bhagat
Review on Security Issues in RFID Systems	Masalah keamanan yang dimiliki RFID	Mohamed El Beqqal, Mostafa Azizi

- Smart Locks: Lessons for Securing Commodity Internet of Things Devices
  Penelitian yang di lakukan oleh Grant Ho dan tim dari University of California, Berkeley ini membahas tentang masalah keamanan yang di temukan di 5 perangkat *smart lock* dengan 4 kemungkinan: physically-present attacker, revoked attacker, thief, dan relay attacker. Physical-present attack di lakukan dengan mengembangkan serangan dari hasil obervasi dari aktifitas pengguna dengan smart lock, revoked attacker di lakukan dengan menggunakan mencari celah yang di dapat dari akses yang lama, thief dilakukan dengan mencuri perangkat yang di gunakan pengguna untuk mengakses smart lock, dan relay attack di lakukan dengan mengeksploit komunikasi smart lock dengan pengguna melalui Bluetooth [5].
- Smart Digital Door Lock System Using Bluetooth Technology
  Paper ini menjelaskan tentang perancangan perangkat *smart lock* dengan menggunakan Bluetooth. Owner atau admin dari *smart lock* ini dapat memantau aktifitas yang terjadi di pintu melalui internet dimana perangkat *smart lock* terbagi menjadi 2, bagian membuka dan mengunci pintu di kendalikan oleh microcontroller AVR atmega32 dan cloud computing untuk memproses data. Smart lock ini memiliki camera yang bisa di gunakan pengguna yang terdaftar untuk memberi akses ke orang lain yang tidak terdaftar [6].
- Review on Security Issues in RFID Systems
  Paper ini menjelaskan tentang permasalahan umum pada keamanan pada RFID dimana ada 4 tipe: Eavesdropping, Denial of Service, Clonning, Tracking. Tujuan utama dari serangan Eavesdropping adalah untuk mengumpulkan data sebanyak mungkin dengan mengintercept komunikasi antara reader dan tags RFID, Denial of Service di tujukan untuk membuat scanner tidak bisa di gunakan lagi dengan contoh mengirimkan sinyal dalam frekuensi tertentu yang dapat membuat reader tidak dapat di gunakan, cloning adalah dengan membuat versi kartu yang baru dengan data yang sama tracking sama halnya dengan cloning, namun pada tracking penyerang akan melakukan autentikasi dengan tag yang dimiliki kartu yang valid dengan kartu yang berbeda [7].
- Raspberry Pi 3 Model B+
  Raspberry Pi 3 Model B+ adalah single-board computer yang di produksi dan di jual oleh Raspberry Pi Foundation beroperasi dengan Central Processing Unit (CPU) Cortex-A53(ARMv8) 64-bit SoC @ 1.4GHz dengan 1GB LPDDR2 SDRAM. Raspberry Pi 3 Model B+ (Pi 3 B+) pada tugas akhir ini menggunakan 16 GB micro SD Card dengan Raspbian OS versi 5.4.79. Pi 3 B+ memerlukan daya sebesar 5V/2,5A DC dengan penggunaan current minimal 500mA. Pi 3 B+ memiliki GPIO (General Purpose Input Output) dengan 30 pin dengan daya output sebesar 3,3V/50mA, 2 pin dengan output 5V yang di dapat dari daya input, dan 8 ground pin dengan total 40 pin [8].
- Time-Based One-Time Password (TOTP)
  One Time Password adalah password yang bisa di gunakan hanya 1 kali yang di dapat dengan membuat nomor acak dengan mengikuti algoritma dalam batas waktu tertentu. Pada tugas akhir ini, penulis menggunakan algoritma Time-Based One-Time Password (TOTP). Time-Based One-Time Password (TOTP) atau RFC 6238 merupakan pengembangan dari HMAC-based One-Time Password (HOTP) atau RFC 4226, dimana pada HOTP password di dapat dengan *truncate* (memotong) nilai HMAC-SHA-1 yang di dapat dari K (Secret) dan C (Counter). $HOTP\ (K, C) = Truncate(HMAC - SHA - 1(K, C)$
  Pada TOTP, nilai C (Counter) di ganti dengan nilai T (Time) menggunakan time reference yaitu UNIX Timestamp [9]. Pada tugas akhir ini PyOTP Library [10] di gunakan untuk menjalankan algoritma TOTP dengan python di dalam Pi 3 B+. Library ini mempunyai 2 fungsi: Pertama membuat secret baru untuk setiap user dengan RFID card masing – masing, Kedua adalah membandingkan kode yang di terima oleh program dari input user dengan kode OTP yang di ketahui oleh library. Jika kode sama maka proses autentikasi valid. Kode OTP pada tugas akhir ini hanya berlaku untuk 30 detik. Library PyOTP sudah memenuhi syarat algoritma poin R1-R6 pada section 3 dari RFC6238 dan untuk memenuhi syarat pada poin R7 penulis hanya memberi izin untuk mengakses data *secret* di database kepada 1 username,

sehingga data yang di gunakan oleh perangkat hanya bisa di akses oleh 1 orang. Alasan pemilihan algortima TOTP di banding dengan menggunakan password biasa adalah kemampuannya untuk membuat password baru secara acak setiap misalkan 30 detik. Keuntungan yang di dapat dengan menggunakan TOTP adalah kemungkinan penyerang melakukan serangan bruteforce password pada sistem sangat kecil.

- RC522

  Modul RFID ini merupakan *bundle* yang terdiri dari reader dan 2 RFID chip (keyfob dan kartu). Cara kerja RC522 adalah *reader* akan memancarkan sinyal elektomagnetik dengan frekuensi 13,56MHz melalui *coil*/antena dan ketika kartu/keyfob dekat dengan reader, *integrated circuit* (IC) pada kartu/keyfob akan berkeasi ke sinyal elektomagnetik dari *coil* pada reader dan mengtransferkan data yang ada di memori ke reader dengan menggunakan SPI interface [11]. Pada tugas akhir ini, penulis menggunakan library MFRC522-python [12] hanya untuk menerima data dari IC di kartu/keyfob. Pemilihan library ini karena penulis hanya memerlukan uid dari RFID chip, dan library MFRC522-python memiliki 1 file yang khusus untuk mendeteksi uid dari RFID. RFID di pilih karena GPIO pada Raspberry Pi hanya bisa mengeluarkan arus DC maksimal 50 mA. Sensor RFID yang di pilih penulis (RC522) hanya menggunakan arus DC maksimal 26 mA, sementara jika menggunakan sensor fingerprint seperti FPM10A [13] menggunakan 120 mA. Penggunaan arus daya lebih dari 50 mA dapat merusak Raspberry Pi.

3. Sistem yang Dibangun

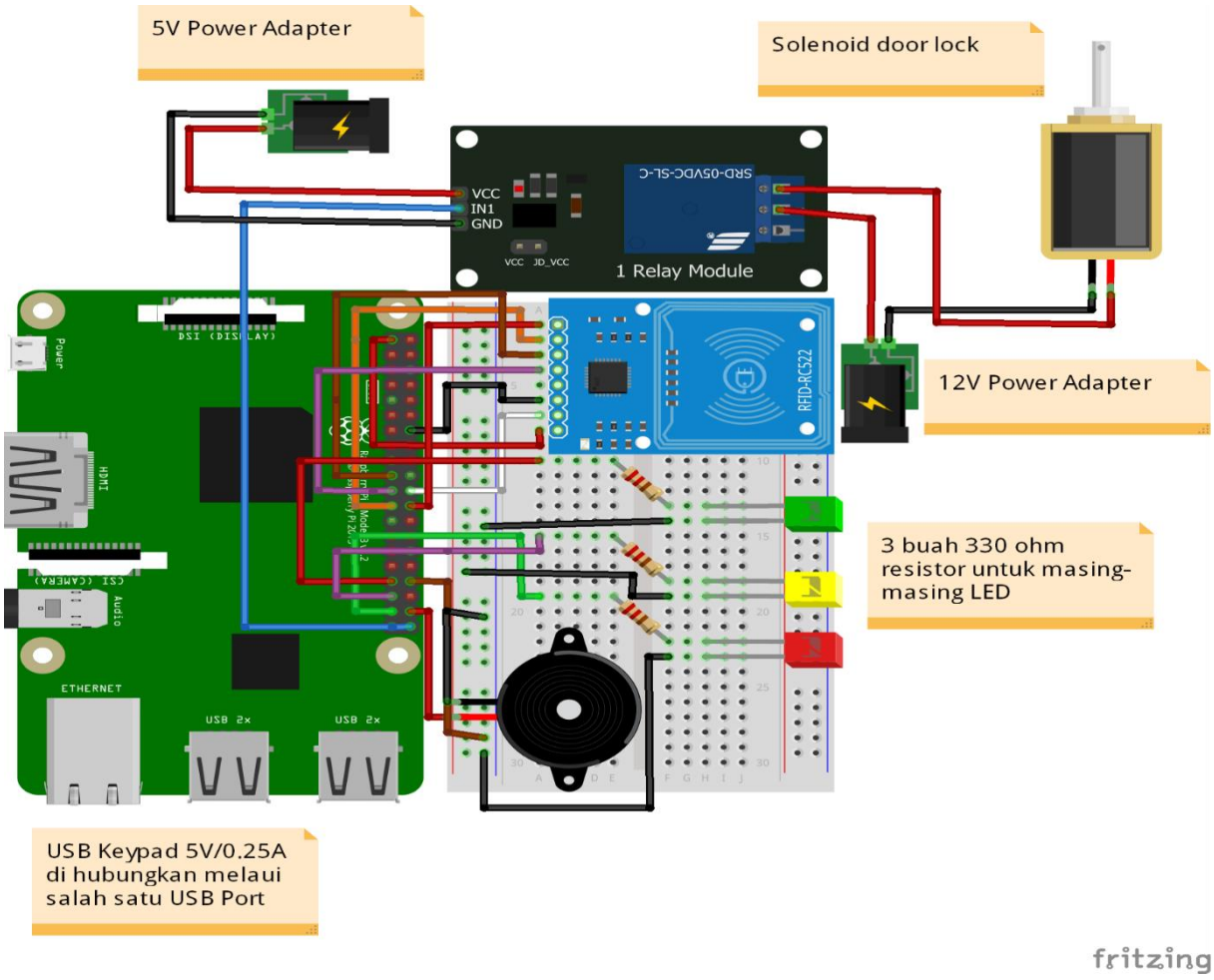
Hasil akhir dari TA ini adalah sebuah perangkat IoT pada *Smart Lock* dengan menggunakan *hardware,software, protocol controller* dan *python library* sebagai berikut:

Hardware
- Raspberry Pi 3 B+ kit
- Generic USB Keypad
- LED merah, kuning, dan hijau
- RC522 RFID sensor kit (sensor dengan 2 RFID berupa keyfob dan card) + 1 RFID card tambahan
- Modul relay 3.3V
- Breadboard
- Jumper wire (F-M, M-M, F-F)
- 3 buah 330 Ω resistor
- Android smartphone
- Personal computer

Software
- Google Authenticator di Android smartphone
- Authy di personal computer
- Mariadb
- Top [14]
- Psutil [15]

Controller
- Serial peripheral interface (SPI)

Pyhton Library
- Spidev
- PyOTP
- MFRC522-python
- Gpiozero



Gambar 1 Rancangan perangkat

Berikut adalah logika dari perangkat *smart lock* dari proses awal membuka kunci hingga kunci terbuka

1. Smart lock posisi idle, RFID scanner memeriksa apakah ada RFID yang bisa di deteksi
2. Scanner mendeteksi ada RFID chip di scanner
   2.1 Jika RFID chip tidak terdaftar di database, kembali ke proses 1
   2.2 Jika RFID chip terdaftar di database, lanjut ke proses 3
3. Program mengambil 3 data dari database berupa nama, secret, dan rfid_uid yang di miliki oleh RFID chip yang terdeteksi
4. Mulai proses autentikasi OTP
   Dengan secret yang di dapat dari proses 3, program membuat kode TOTP dan mengassign kode OTP sebagai variable OTP
5. Autentikasi OTP
   Pengguna mengetikan kode OTP yang di dapat dari aplikasi Google Authenticator atau Authy
   5.1 Jika kode salah, kembali ke proses 4 dan ulang proses hingga proses 5. Proses 5.1 di lakukan maksimal 3 kali. Jika lebih dari 3 kali, program reset dan kembali ke proses 1
   5.2 Jika kode benar, lanjut ke proses 6
6. Program membuka kunci, reset. Kembali ke proses 1

Karena *smart lock* di rancang untuk berjalan secara *offline*, untuk mendaftarkan/mengubah data kartu RFID di sistem, admin perlu menghubungkan Raspberry Pi dengan monitor, keyboard, dan mouse. Database yang menyimpan data berupa nama pengguna, rfid pengguna, dan *secret* hanya bisa di akses oleh 1 akun. Waktu yang di gunakan pada Raspberry Pi adalah mengikuti timezone yang sesuai dengan dimana perangkat di pasang, dengan mengikuti UNIX timestamp.

Perangkat akan di analisa dari segi fungsional, akurasi deteksi, *code response* , keamanan, dan *overhead*. Berikut adalah skenario pelaksanaan bagaimana perangkat akan di analisa:

Uji fungsional

Tujuan dari pengujian ini adalah memastikan bahwa sistem dapat menjalankan program yang dapat mengatur semua koneksi ke masing – masing perangkat komponen dengan baik dengan 3 kartu RFID yang berbeda dan 2 aplikasi TOTP yang berbeda.

1. Uji fungsional dengan Google Authenticator
   - 1.1 Mendaftarkan 3 kartu
     3 kartu yang di sediakan di daftarkan ke perangkat dengan menjalankan "save-user.py" dan secret yang di dapat akan di inputkan ke Google Authenticator dengan nama kartu Blue Dongle (kartu 1), Ruber White (kartu 2), dan Basic White (kartu 3). Blue Dongle dan Ruber White adalah kartu yang terdaftar ke sistem, Basic White adalah kartu yang tidak terdaftar ke sistem.
   - 1.2 Menjalankan program periksa kartu
     Jalankan "check-user.py" di perangkat secara manual.
   - 1.3 Proses autentikasi
     Tempelkan RFID card dengan nama Blue Dongle ke modul RFID.
     - 1.3.1 Jika alarm berbunyi, maka RFID tersebut belum / tidak terdaftar. Ulang proses dari 1.1
     - 1.3.2 Jika LED merah mati dan LED kuning hidup, ketikan kode yang tampil di aplikasi Google Authenticator dari yang bernama Blue Dongle.
       - 1.3.2.1 Jika alarm berbunyi, maka kode salah. Ulang proses hingga 3 kali. Jika masih sama maka kode tidak akurat dan tes ini gagal.
       - 1.3.2.2 Jika solenoid door lock aktif ke arah membuka kunci, LED kuning mati dan LED hijau hidup, maka kode benar dan tes ini berhasil.
   - 1.4 Ulangi 1.3 dengan kartu bernama Ruber White dan Basic White. Khusus untuk Basic White dimana kartu ini adalah kartu yang tidak di daftarkan, proses berakhir di 1.3.1
2. Uji fungsional dengan Authy
   - 2.1 Mendaftarkan 3 kartu
     Karena perangkat memiliki database, maka tidak perlu di lakukan pendaftaran kartu lagi. 3 kartu yang sebelumnya di gunakan di atas bisa di gunakan lagi dengan menggunakan secret yang sama. Masukkan secret dari database ke Authy dengan masing-masing nama nama Blue Dongle, Ruber White, dan Basic White. Blue Dongle dan Ruber White tetap menjadi kartu yang terdaftar ke sistem.
   - 2.2 Menjalankan program periksa kartu
     Jalankan "check-user.py" di perangkat.
   - 2.3 Tempelkan RFID card dengan nama Blue Dongle.
     - 2.3.1 Jika alarm berbunyi, maka RFID tersebut belum / tidak terdaftar. Ulang proses dari 2.1
     - 2.3.2 Jika LED merah mati dan LED kuning hidup, ketikan kode yang tampil di aplikasi Google Authenticator dari yang bernama Blue Dongle..
       - 2.3.2.1 Jika alarm berbunyi, maka kode salah. Ulang proses hingga 3 kali. Jika masih sama maka kode tidak akurat dan tes ini gagal.
       - 2.3.2.2 Jika solenoid door lock aktif ke arah membuka kunci, LED kuning mati dan LED hijau hidup, maka kode benar dan tes ini berhasil.
   - 2.4 Ulangi 1.3 dengan kartu bernama Ruber White dan Basic White. Khusus untuk Basic White dimana kartu ini adalah kartu yang memang tidak di daftarkan, sehingga proses akan berakhir di 2.3.1

Akurasi Deteksi

Tujuan dari pengujian ini adalah memastikan fungsi autentikasi dalam program dapat mendeteksi kode yang benar. Kode invalid menggunakan data yang di dapat oleh Ding et al [16] yang memiliki tingkat persentasi yang tinggi. Berikut adalah skenario pengerjaan pengujian:

1. Pemilihan kartu RFID
   Pengujian ini menggunakan kartu valid yang sudah di daftarkan pada pengujian fungsional (Blue Dongle dan Ruber White).
2. Menjalankan program periksa kartu
   Jalankan "check-user.py" di perangkat.
3. Proses autentikasi
   - 3.1 Autentikasi dengan Google Authenticator
     Tempelkan kartu RFID dengan nama Ruber White ke modul RFID.
     - 3.1.1 Autentikasi kode invalid

Kode yang di gunakan pada pengujian ini adalah “123456” dan “111111”

3.1.1.1 Ketikan kode “123456”
Jika alarm berbunyi lanjut ke proses selanjutnya, jika benar maka sistem tidak akurat dalam mendeteksi kode. Pengujian ini gagal

3.1.1.2 Ketikan kode “111111”
Jika alarm berbunyi lanjut ke proses selanjutnya, jika benar maka sistem tidak akurat dalam mendeteksi kode. Pengujian ini gagal

3.1.1.3 Ketikan kode yang di dapat dari Google Authenticator
Jika alarm berbunyi, maka sistem tidak akurat dalam mendeteksi kode, sehingga pengujian ini gagal. Jika benar maka program dapat mendeteksi kode yang valid, pengujian ini berhasil.

3.2 Autentikasi dengan Authy
Tempelkan kartu RFID dengan nama Ruber White ke modul RFID

3.2.1 Autentikasi kode invalid
Kode yang di gunakan pada pengujian ini adalah “123456” dan “111111”

3.2.1.1 Ketikan kode “123456”
Jika alarm berbunyi lanjut ke proses selanjutnya, jika benar maka sistem tidak akurat dalam mendeteksi kode. Pengujian ini gagal

3.2.1.2 Ketikan kode “111111”
Jika alarm berbunyi lanjut ke proses selanjutnya, jika benar maka sistem tidak akurat dalam mendeteksi kode. Pengujian ini gagal

3.2.1.3 Ketikan kode yang di dapat dari Authy
Jika alarm berbunyi, maka sistem tidak akurat dalam mendeteksi kode, sehingga pengujian ini gagal. Jika benar maka program dapat mendeteksi kode yang valid, pengujian ini berhasil.

Code Response

Dengan menggunakan built-in library “time”, penulis membuat timer mulai dari kartu terdeteksi hingga kunci terbuka. Formula untuk mendapatkan hasil pada pengujian ini adalah dengan $Code\ Response = \frac{\text{rata – rata dari total waktu di eksekusi}}{jumlah\ banyak\ pengujian}$. Program “check_user.py” di ubah menjadi “code_response_cu.py” dan program “check_user_na.py” di ubah menjadi “code_response_cu_na.py”. Kartu yang di gunakan adalah kartu yang valid, yaitu Blue Dongle dan Ruber White dan kartu yang tidak valid, Basic White. Penulis menentukan lama waktu pengetikan kode OTP sepanjang 6 digit adalah selama 2 detik.

Keamanan

Tujuan dari pengujian ini adalah memastikan perangkat *smart lock* mempunyai sistem keamanan dasar yang bisa di eksploit oleh penyerang. Keamanan yang di uji dari tugas akhir ini adalah keamanan pada RFID dan TOTP. Di pilih RFID dan TOTP sebagai fokus pengujian keamanan adalah karena 2 fitur ini adalah fitur utama pada tugas akhir ini

- RFID

Dengan mengutip paper “Review on Security Issues in RFID Systems” [7], penulis menentukan serangan yang bisa terjadi adalah berupa Denial of Service (DoS) dan Cloning. Serangan DoS dapat terjadi saat penyerang hanya melakukan taping dan tidak melakukan autentikasi, sehingga program tidak bisa menyelesaikan proses dan perangkat tidak bisa di gunakan. Serangan cloning dapat terjadi dengan penyerang membuat salinan RFID yang valid ke kartu yang baru dan melakukan taping dengan kartu yang baru tanpa sepengetahuan admin.

- TOTP

Dengan mengutip CVE-2015-7225 [17] dimana di temukan celah pada TOTP. Celah ditemukan dengan menggunakan ulang kode valid yang sudah di gunakan oleh user. Hal ini terjadi karena program Two Factor Authentication oleh Tinfoil Security ini masih bisa menerima kode valid yang seharusnya kadaluarsa setelah di gunakan.

Overhead

Pengujian ini di lakukan karena untuk mencari data seberapa banyak program menggunakan sumber daya CPU dan RAM di sistem. Raspberry Pi 3 B+ di tenagai dengan CPU jenis BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC dengan kecepatan 1,4Ghz dan memiliki 1GB LPDDR2 SDRAM.

Untuk memastikan data yang di dapat akurat, pada pengujian overhead ini penulis menggunakan 2 program untuk merekam aktivitas "check_user.py" dan "check_user_na.py" dalam penggunaan CPU dan RAM. Yaitu dengan "top" dan "psutil".

"top" adalah program bawaan dari Raspbian OS yang di gunakan untuk mengawasi aktivitas di sistem seperti CPU, RAM, dan jaringan yang di buat menggunakan Bahasa C. "psutil" adalah *python library* yang bisa di gunakan untuk mengawasi aktivitas di sistem seperti "top". Namun pengguna harus membuat programnya terlebih dahulu. Karena "top" dan "psutil" di buat dengan 2 bahasa pemogrman yang berbeda, alasan ini lah penulis memilih 2 program ini untuk pengujian overhead ini. Di sini penulis membuat program dengan nama" psutil_check_user.py" dan "psutil_check_user_na.py".

a) Validasi program

Pengujian ini di lakukan untuk melihat apakah 2 program yang di pilih dapat memastikan hasil yang di peroleh bisa di validasi kebenarannya 1 sama lain, hal ini di karenakan data yang di peroleh dapat di pastikan akurat. Pengujian di lakukan dengan mengukur seberapa banyak program pada tugas akhir ini menggunakan sumber daya dari CPU dan RAM. Proses yang di perhatikan adalah "check_user.py" dan "check_user_na.py" dengan menggunakan program "top" dan "psutil". Berikut adalah scenario pengerjaannya:

1. Dengan "top"
   1.1. Jalankan program "check_user.py" di perangkat
   1.2. Simpan seberapa banyak "check_user.py" menggunakan CPU dan RAM
      1.2.1 Pada idle awal.
         Simpan data dengan menggunakan command "top -b -u pi -n 10 | grep python > log_check_user_idle1.txt"
      1.2.2 Pada eksekusi fungsi.
         Lakukan taping dengan kartu RFID yang valid dan simpan data dengan menggunakan command "top -b -u pi -n 10 | grep python > log_check_user_eksekusi_fungsi.txt".
      1.2.3 Pada idle setelah eksekusi fungsi.
         Lakukan autentikasi dan tunggu hingga LED merah menyala lagi, simpan data dengan menggunakan command "top -b -u pi -n 10 | grep python > log_check_user_idle2.txt".
   1.3. Jalankan program "check_user_na.py" di perangkat
   1.4. Simpan seberapa banyak "check_user_na.py" menggunakan CPU dan RAM. Simpan data dengan menggunakan command "top -b -u pi -n 10 | grep pyhton > log_check_user_na.txt". Tidak perlu melakukan taping.

Untuk program "check_user_na.py" pada top tidak di lakukan proses perekaman pada saat eksekusi fungsi karena tidak di mungkinkan, lebih detailnya di bahas di analisa. Berikut adalah penjelasan *command* yang di gunakan saat menjalankan program "top".

- b : menggunakan mode *batch* sehingga program bisa memberi sebuah output yang bisa di simpan
- u : memilih user mana yang akan di fokuskan, dalam hal ini adalah "pi" karena pada user ini penulis melakukan pengujian.
- n : berapa kali proses ini di lakukan dengan jeda 1 iterasi (2,64 detik) per proses. Dalam hal ini 10 iterasi (26,4 detik)
- grep : karena user "pi" (username yang di gunakan penulis di perangkat) tidak hanya menjalankan program pada tugas akhir ini, command ini di gunakan untuk hanya menampilkan apa saja yang terkait dengan "python" pada user "pi"

2. Dengan "Psutil"
   2.1. Catat PID
      Jalankan program "check_user.py" di perangkat dan catat Process Identification pada "check_user.py" dengan "top". *Proses Identification* (PID) adalah ID yang di berikan dari sistem ke setiap proses yang sedang berjalan / di gunakan di dalam sistem. "top" di gunakan kembali untuk mencatat PID apa yang di miliki oleh "check_user.py".

2.2. Buat program “psutil_check_user.py” dengan library psutil

Psutil adalah python library yang berisikan fungsi – fungsi yang bisa di gunakan untuk memantau dan merekam aktivitas CPU, RAM, penyimpanan, dan jaringan di dalam perangkat. Berikut adalah *code* yang di gunakan penulis untuk “psutil_check_user.py”.

```
import psutil
import time

p = psutil.Process(2121) #Ubah 2121 sesuai PID untuk check_user.py
n = 0

try:
    while n!= 10:
        print("CPU yang digunakan: ", p.cpu_percent(interval = 2.64))
        print("Memory yang digunakan: ", p.memory_percent())
        print("\n")
        n = n + 1
        time.sleep(1)
except KeyboardInterrupt:
    print("quit")
```

Gambar 2 *Code* untuk psutil pada pengujian overhead

2.3. Jalankan program “psutil_check_user.py” dan simpan data untuk proses idle awal, eksekusi fungsi, dan idle setelah eksekusi fungsi untuk “check_user.py”

2.3.1. Pada idle awal

Jalankan program melalui terminal dengan *command* “python3 psutil_check_user.py > result_check_user_idle1.txt”.

2.3.2. Pada eksekusi fungsi

Tap kartu yang valid dan jalankan program melalui terminal dengan *command* “python3 psutil_check_user.py > result_check_user_eksekusi_fungsi.txt”.

2.3.3. Pada idle setelah eksekusi fungsi

Lakukan autentikasi dan tunggu LED merah untuk menyala. Jalankan program melalui terminal dengan *command* “python3 psutil_check_user.py > result_check_user_idle2.txt”.

2.4. Catat PID

Jalankan program “check_user_na.py” di perangkat dan catat PID pada “check_user_na.py” dengan “top”.

Buat program “psutil_check_user_na.py”

Ulangi proses 2.1 dengan program “check_user_na.py” dan catat PID yang terkait dengan “check_user_na.py”

2.5. Buat program “psutil_check_user_na.py”

Dengan menggunakan *code* yang sama dengan 2.2, ganti PID dengan PID untuk “check_user_na.py”

2.6. Jalankan program “psutil_check_user_na.py” dan simpan data untuk proses idle untuk “check_user_na.py”

2.6.1. Rekam posisi idle

Jalankan program melalui terminal dengan *command* “python3 psutil_check_user.py > result_check_user_na.txt”.

Sama dengan pada pengujian dengan “top”, proses eksekusi fungsi pada “check_user_na.py” tidak di rekam karena agar data yang di dapat bisa sama. Berikut adalah penjelasan singkat dari kode yang di gunakan di dalam program.

- cpu_percent : Mengembalikan nilai penggunaan CPU dalam bentuk persen
- memory_percent : Mengembalikan nilai penggunaan RAM dalam bentuk persen

b) Perbandingan check_user.py dan check_user_na.py

Untuk mengukur perbandingan penggunaan sumber daya pada sistem, psutil di pilih oleh penulis karena kemampuannya untuk mengkur hingga ke milidetik. Kartu yang pilih adalah Ruber White. Selama 15 detik, program akan memberikan output berupa penggunaan CPU dalam bentuk persen setiap 0,1 detik, dengan jarak 0.1 detik. Berikut adalah *code* untuk pengukuran CPU.

```
import psutil
import time

p = psutil.Process("PID")
t_end = time.time() + 15

try:
    while time.time() < t_end:
        print(p.cpu_percent(interval=0.1))
        time.sleep(0.1)

except KeyboardInterrupt:
    print("quit")
```

Gambar 3 *Code* untuk melakukan pengukuran data CPU pada pengujian overhead

Untuk pengukuran pada RAM, pengujian juga di lakukan selama 15 detik dengan perbedaan program akan mengouputkan data berupa penggunaan RAM setiap 0,1 detik. Tidak ada *interval* untuk pengukuran RAM. Berikut adalah *code* untuk pengukuran RAM.

```
import psutil
import time

p = psutil.Process("PID")
t_end = time.time() + 15

try:
    while time.time() < t_end:
        print(p.memory_percent())
        time.sleep(0.1)

except KeyboardInterrupt:
    print("quit")
```

Gambar 4 Code untuk melakukan pengukuran data RAM pada pengujian overhead

4. Evaluasi

4.1 Hasil Pengujian

1. Uji fungsional

Tabel 2 Hasil pengujian fungsional dengan kode dari Google Authenticator

	Blue Dongle	**White Ruber**	**Basic White**
Perangkat mendeteksi kartu	Yes	Yes	No
Kode valid	Yes	Yes	-
Hasil sesuai skenario	100%	100%	100%

Tabel 3 Hasil pengujian fungsional dengan kode dari Authy

	Blue Dongle	**White Ruber**	**Basic White**
Perangkat mendeteksi kartu	Yes	Yes	No
Kode valid	Yes	Yes	-
Hasil sesuai skenario	100%	100%	100%

2. Akurasi Deteksi

Tabel 4 Hasil pengujian akurasi deteksi dengan kode dari Google Authenticator

Nama kartu	**Dengan kode "123456"**	**Dengan kode "111111"**	**Dengan kode dari aplikasi**
Blue Dongle	Salah	Salah	Benar
White Ruber	Salah	Salah	Benar
Hasil sesuai skenario	100%	100%	100%

Tabel 5 Hasil pengujian akurasi deteksi dengan Authy

Nama kartu	**Dengan kode "123456"**	**Dengan kode "111111"**	**Dengan kode dari aplikasi**
Blue Dongle	Salah	Salah	Benar
White Ruber	Salah	Salah	Benar
Hasil sesuai skenario	100%	100%	100%

3. Code Response

Pengujian ini akan dilakukan sebanyak 10 kali setiap program selesai. Program "code_response_cu.py" di mulai dari kartu terdeteksi hingga membuka kunci jika RFID terdeteksi, jika RFID tidak terdeteksi pengujian di mulai saat kartu terdeteksi hingga program mengeluarkan output bahwa kartu tidak terdeteksi. Pada program "code_response_cu_na.py" proses di mulai dari RFID terdeteksi dan kunci terbuka, jika RFID tidak terdeteksi, pengujian di mulai saat kartu terdeteksi hingga program mengeluarkan output bahwa kartu tidak terdeteksi. Pada pengujian ini, Ruber White adalah kartu valid dan Basic White adalah kartu tidak valid.

Tabel 6 Hasil pengujian code response

Proses	**code_response_cu.py**			**code_response_cu_na.py**	
	Kartu valid(detik)	**Kartu valid setelah dikurang asumsi pengetikan otp selama 2 detik**	**Kartu tidak valid (detik)**	**Kartu valid (detik)**	**Kartu tidak valid (detik)**
1	4,51	2,51	0,00031	0,00054	0,00035
2	4,16	2,16	0,00033	0,00037	0,00037
3	5,39	3,39	0,00031	0,00038	0,00037
4	6,05	4,05	0,00029	0,00038	0,00031
5	5,01	3,01	0,00031	0,00037	0,00035
6	4,42	2,42	0,00031	0,00037	0,00031
7	5,11	3,11	0,00031	0,00037	0,00031
8	6,77	4,77	0,0003	0,00038	0,00031
9	5,23	3,23	0,00032	0,00038	0,00031
10	4,66	2,66	0,00032	0,00037	0,00031
Hasil	**5,131**	**3,131**	**0,000311**	**0,00039**	**0,00033**

4. Keamanan

- RFID

Untuk menangani serangan DoS, di perlukan timer atau time out pada saat proses autentikasi TOTP. Karena kode OTP kadaluarsa dalan waktu 30 detik, maka waktu yang di butuhkan untuk perangkat menjadi timeout adalah 30 detik. Tabel 7 adalah hasil pengujian timeout.

Tidak di lakukannya pengujian untuk cloning RFID karena cara mengatasi cloning RFID hanya dengan mengandalkan pengguna untuk menjaga kartu RFID mereka dengan baik.

Tabel 7 Hasil pengujian keamanan pada RFID

Nama kartu	**Terdeteksi scanner saat taping**	**Time out pada saat lebih dari 30 detik**
Blue Dongle	Iya	Iya
White Ruber	Iya	Iya
Basic White	Tidak	-
Hasil sesuai skenario	100%	100%

- TOTP

Pengujian ini di lakukan melaui analisa pasif pada kode di dalam program. Dalam melakukan proses autentikasi kode OTP, melakukan perbandingan kode yang di terima dari input user dengan isi dari variable TOTP.

```
cursor.execute("Select secret From users Where rfid_uid="+str(id))
secret = result[2]
# Proses ini mengambil data secret dari database dan mengubahnya ke string dan
# mengssign data secret sebagai isi dari variabel secret

totp = pyotp.TOTP(str(secret))
# membuat variable dengan nama totp
# yang berisi data dari variabel secret
# agar bisa di pahami oleh library

otp = totp.now()
# Membuat kode TOTP
```

Gambar 5 Proses pembuatan kode TOTP

Kode TOTP hanya di assign ke variable otp, tidak di simpan. Isi dari variable otp akan di reset setiap proses autentikasi totp di lakukan, baik autentikasi berhasil atau tidak. Dengan ini kode “kadaluarsa” dan tidak bisa di gunakan lagi jika sudah melebihi batas waktu 30 detik yang di tetapkan.

5. Overhead

a) Validasi program

TOP

Dengan menjalankan program “top” dengan command “top -b -u pi -n 10 | grep python > log_check_user.txt” untuk check user dan command “top -b -u pi -n 10 | grep python > log_check_user_na.txt”

Tabel 8 Hasil pengujian overhead dengan TOP untuk program check_user.py

check_user.py						
Proses	Penggunaan CPU (%)			Penggunaan RAM (%)		
	Idle awal	**Eksekusi fungsi**	**Idle setelah eksekusi fungsi**	**Idle awal**	**Eksekusi fungsi**	**Idle setelah eksekusi fungsi**
1	88,2	0,1	88,2	1,6	1,6	1,6
2	85,8	0,3	88	1,6	1,6	1,6
3	86,4	0,1	88,1	1,6	1,6	1,6
4	89,1	0,3	89,1	1,6	1,6	1,6
5	89,7	0,3	87,7	1,6	1,6	1,6
6	88,4	0,1	88,4	1,6	1,6	1,6
7	88,1	0,3	88,7	1,6	1,6	1,6
8	88,4	0,7	87,7	1,6	1,6	1,6
9	89,4	0,3	87,4	1,6	1,6	1,6
10	88,4	0,1	87,4	1,6	1,6	1,6
Rata -Rata	**88,19**	**0,26**	**88,07**	**1,6**	**1,6**	**1,6**

Tabel 7 di atas adalah detail dari hasil yang di dapat dengan menggunakan program “top”. Tabel 8 di bawah ini adalah hasil rata – rata dari tabel 7.

Tabel 9 Rata – rata hasil pengujian overhead dengan TOP untuk program check_user.py

check_user.py		
Iterasi ke	**Rata – rata penggunaan CPU(%) pada program di posisi idle pertama, eksekusi fungsi di program, idle setelah program selesai eksekusi fungsi**	**Rata – rata penggunaan RAM(%) pada program di posisi idle pertama, eksekusi fungsi di program, idle setelah program selesai eksekusi fungsi**
1	**58,33**	**1,6**
2	**58,03**	**1,6**
3	**58,2**	**1,6**
4	**59,5**	**1,6**
5	**59,23**	**1,6**
6	**58,96**	**1,6**
7	**59,03**	**1,6**
8	**58,93**	**1,6**
9	**59,03**	**1,6**
10	**58,63**	**1,6**
Rata – rata	**58,84**	**1,6**

Tabel 10 Hasil pengujian overhead dengan TOP untuk program check_user_na.py

check_user_na.py		
Iterasi ke	**CPU(%) pada posisi idle**	**RAM(%) pada posisi idle**
1	83,3	1,6
2	89,1	1,6
3	88,4	1,6
4	87,7	1,6
5	87,7	1,6
6	87,1	1,6
7	87,7	1,6
8	87,4	1,6
9	87,7	1,6
10	87,4	1,6
Rata - rata	**87,35**	**1,6**

Data yang di peroleh dari “check_user.py” dan “check_user_na.py” berbeda karena pada program “check_user_na.py” tidak ada autentikasi, sehingga tidak di mungkinkan untuk menggunakan top untuk merekam penggunaan CPU dan RAM mengingat dengan melihat hasil yang di dapat dari pengujian *code response* untuk “check_user_na.py”, fungsi periksa selesai dalam milidetik.

Psutil

Tabel 11 Hasil pengujian overhead dengan psutil untuk program check_user.py

check_user.py						
Proses	Penggunaan CPU (%)			Penggunaan RAM (%)		
	Idle awal	**Eksekusi fungsi**	**Idle setelah eksekusi fungsi**	**Idle awal**	**Eksekusi fungsi**	**Idle setelah eksekusi fungsi**
1	89,7	0	88,9	1,6	1,6	1,6
2	89,3	0,4	87,4	1,6	1,6	1,6
3	90,4	0,4	92,3	1,6	1,6	1,6
4	88,2	0,4	86,6	1,6	1,6	1,6
5	88,5	0	89,3	1,6	1,6	1,6
6	88,2	0	90	1,6	1,6	1,6
7	87,8	0	88,2	1,6	1,6	1,6
8	88,2	0,4	89,3	1,6	1,6	1,6
9	87,4	0,4	88,5	1,6	1,6	1,6
10	87,8	0,4	91,6	1,6	1,6	1,6
Rata -Rata	**88,55**	**0,24**	**89,21**	**1,6**	1,6	**1,6**

Tabel 10 di atas adalah detail dari hasil yang di dapat dengan menggunakan program “top”. Tabel 11 di bawah ini adalah hasil rata – rata dari tabel 10.

Tabel 12 Rata – rata hasil pengujian overhead dengan psutil untuk program check_user.py

check_user.py		
Iterasi ke	**Rata – rata penggunaan CPU(%) pada program di posisi idle pertama, eksekusi fungsi di program, idle setelah program selesai eksekusi fungsi**	**Rata – rata penggunaan RAM(%) pada program di posisi idle pertama, eksekusi fungsi di program, idle setelah program selesai eksekusi fungsi**
1	**59,56**	**1,6**
2	**59,03**	**1,6**
3	**61,03**	**1,6**
4	**58,4**	**1,6**
5	**59,3**	**1,6**
6	**59,43**	**1,6**
7	**58,7**	**1,6**
8	**59,3**	**1,6**
9	**58,76**	**1,6**
10	**59,93**	**1,6**
Rata – rata	**59,34**	**1,6**

Berikut adalah tabel 12 yang berisi tentang hasil pengujian overhead pada “check_user_na.py”. Tidak di lakukan perekaman saat eksekusi fungsi karena sama dengan pada “top”, perekaman di lakuan dengan interval 2,64 detik.

Tabel 13 Hasil pengujian overhead dengan psutil untuk program check_user_na.py

check_user_na.py		
Iterasi ke	**CPU(%) pada posisi idle**	**RAM(%) pada posisi idle**
1	90,1	1,6
2	90,1	1,6
3	90,1	1,6
4	88,5	1,6
5	88,9	1,6
6	88,2	1,6
7	88,2	1,6
8	89,3	1,6
9	88,9	1,6
10	88,5	1,6
Rata - rata	**89,08**	**1,6**

b) Perbandingan

Tabel di bawah ini adalah hasil rata – rata yang di dapat dengan pengujian untuk perbandingan overhead. Data yang di peroleh berisikan 75 data CPU dan 75 data RAM pada "check_user.py" dan 75 data CPU dan 75 data RAM pada "check_user_na.py". Data di peroleh dengan lama perekaman 15 detik.

Tabel 14 Rata - rata dari hasil pengujian perbandingan untuk overhead pada check_user dan check_user_na

check_user		**check_user_na**	
Rata – rata penggunaan CPU dalam persen	**Rata – rata penggunaan RAM dalam persen**	**Rata – rata penggunaan CPU dalam persen**	**Rata – rata penggunaan RAM dalam persen**
42,823	1,599	83,543	1,5993

4.2 Analisis Hasil Pengujian

1. Uji Fungsional

   Pada sisi program, perangkat dapat menjalankan semua fungsi dengan baik. Program dapat mengidentifikasi antara kartu RFID yang valid dan tidak valid, program dapat memeriksa apakah kode yang di input oleh user benar, dan semua feedback berupa LED dan buzzer berfungsi dengan baik jika di perlukan.

   Ada beberapa kejadian perangkat menolak kode saat menggunakan kode 5 detik baru di buat atau 5 detik akan kadaluarsa. Hal ini di sebabkan program hanya menerima kode dalam batas waktu 30 detik dengan format waktu UTC, karena itu perangkat batas maksimal 3 kali untuk ketik kode ulang. Hal ini umumnya dapat di atasi dengan merancang perangkat agar bisa menerima kode 30 detik yang lalu dan juga 30 detik yang akan datang, sehingga memberi batas waktu selama 90 detik. Penulis tidak menerapkan ini karena kurangnya pengetahuan dalam menggunakan library PyOTP secara maksimal.

2. Akurasi deteksi

   Dengan menggunakan data dari penelitian "Understanding Human-Chosen PINs:Characteristics, Distribution and Security" [16] sebagai kode otp yang salah, dapat di simpukan bahwa perangkat dapat mendeteksi kode yang benar dan yang salah. Karena sistem berjalan dengan offline, sistem tidak mungkin untuk di serang dengan cara *bruteforce* pin otp melalui internet. Satu – satunya cara untuk melakukan *bruteforce* pin otp adalah dengan cara manual, namun karena kode hanya valid selama 30 detik, kemungkinan untuk mendapatkan kode yang benar adalah sekitar 15/46,656 (panjang pin adalah 6, sehingga 6^6 = 46,656 dengan asumsi waktu pengetikan kode 2 detik).

3. Code response

   Pengujain ini di lakukan untuk mengetahui seberapa lama di perlukan untuk menyelesaikan semua fungsi dari awal hingga akhir dengan asumsi di perlukan 2 detik untuk mengetikkan kode OTP. Di perlukan sekitar 3,131 detik untuk memproses semua fungsi untuk "check_user.py" dan sekitar 0,39 milidetik untuk memproses semua fungsi pada "check_user_na.py" dengan kartu valid. Pada kartu tidak valid hanya di perlukan 0,31 mili detik untuk "check_user.py" dan 0,033 milidetik untuk "check_user_na.py" kartu tidak valid. Sehingga perbandingannya dapat di lihat pada tabel 14.

Tabel 15 Perbandingan lama penyelesaian program untuk pengujian code response

Perbandingan lama penyelesaian program dengan kartu valid antara check_user.py dan check_user_na.py	**Perbandingan lama penyelesaian program dengan kartu tidak valid antara check_user.py dan check_user_na.py**
3,0325 detik	0,278 milidetik

4. Keamanan

Program dapat mendeteksi jika proses input kode OTP lebih 30 detik dan menjalankan fungsi timeout dan reset program kembali ke posisi idle awal. Seperti yang sudah di singgung di hasil pengujian, penulis tidak melakukan pengujian untuk mengatasi serangan cloning RFID karena serangan itu hanya terjadi jika pengguna tidak menjaga RFID dengan baik. Namun, dengan adanya 2 step authentication walaupun penyerang berhasil melakukan cloning RFID, penyerang masih harus mendapatkan kode OTP.

Di seksi 5 poin ke 5.2 pada dokumen RFC 6238 menjabarkan bahwa kode harus hanya bisa gunakan 1 kali. Dengan melakukan pasif analisa melalui kode di program, kode OTP hanya di simpan sebagai isi dari variable dan akan terus berubah setiap permintaan autentikasi di lakukan. Hal ini memastikan kode yang di gunakan tidak sama dan menjamin hanya bisa di gunakan *one time*.

5. Overhead

a) Validasi program

Pengujian overhead menggunakan 2 program top dan psutil. Digunakan 2 program untuk pengujian overhead karena untuk membandingkan hasil agar data bisa di bandingkan satu sama lain untuk di validasi. Penulis melampirkan hasil pengujian lebih detail di lampiran. Data yang di dapat dari top jika di bandingkan dengan psutil hanya memiliki selisih yang kecil, sehingga penulis dapat menyimpulkan bahwa data yang di dapat dari top dapat di validasi kebenarannya dan begitu pula sebaliknya. Tabel 15 di bawah ini adalah rata – rata perbandingan selisih dari program "check_user.py" dan tabel 16 adalah rata – rata perbandingan selisih dari program "check_user_na.py".

Tabel 16 Selisih dari data yang di peroleh dari top dan psutil pada program check_user.py

Rata – rata selisih penggunaan CPU(%)			**Rata – rata selisih penggunaan RAM (%)**		
Idle awal	Eksekusi fungsi	Idle setelah eksekusi fungsi	Idle	Eksekusi fungsi	Idle setelah eksekusi fungsi
0,4	**0**	**1,1**	**0,0**	**0,0**	**0,0**

Tabel 17 Selisih dari data yang di peroleh dari top dan psutil pada program check_user_na.py

Rata – rata selisih penggunaan CPU(%)			**Rata – rata selisih penggunaan RAM (%)**		
Idle	Eksekusi fungsi	Idle setelah eksekusi fungsi	Idle	Eksekusi fungsi	Idle setelah eksekusi fungsi
1,73	**-**	**-**	**0,0**	**-**	**-**

Dengan hasil yang di dapat dari top dan psutil, penulis menemukan bahwa penggunaan CPU yang tinggi pada saat idle awal dan idle setelah eksekusi fungsi di program "check_user.py" dan "check_user_na.py". Penggunaan CPU yang tinggi pada saat idle di 2 program di akibatkan karena perangkat terus menerus melakukan scanning tanpa jeda [18]. Library yang penulis gunakan untuk melakukan *scan* kartu RFID tidak di lengkapi jeda waktu untuk *reader* saat bekerja, sehingga menggunakan banyak sumber daya yang ada di CPU. Penggunaan sumber daya langusng turun ke angka 0,26% saat eksekusi proses karena semua fungsi di kerjakan secara bertahap.

b) Perbandingan

Penggunaan RAM pada kedua program hanya berbeda sebanyak 0,006%. Hal ini bisa di karenakan "check_user_na.py" tidak memiliki fitur TOTP, karena itu tidak banyak memori yang di gunakan di dalam RAM.

Penggunaan CPU berbeda sebanyak 40,72% dengan "check_user.py" memiliki hasil rata – rata yang lebih kecil dari "check_user_na.py" walaupun memiliki fitur TOTP di bandingkan dengan "check_user_na.py" yang hanya menggunakan RFID. Penulis menyimpulkan bahwa hasil yang di dapat dari "check_user.py" lebih baik dari "check_user_na.py" karena "check_user.py" membutuhkan waktu lebih lama untuk menyelesaikan fungsi dari "check_user_na.py". Berikut adalah penjabarannya

- Pengaruh waktu
  Dengan menggunakan data yang di dapat dari *code response*, "check_user.py" membutuhkan sekitar 3,131 detik untuk menyelesaikan semua fungsi dan perintah. Sementara "check_user_na.py" hanya membutuhkan waktu 0,000391 detik.
- Data yang di peroleh
  Dengan mengutip data yang di peroleh dari validasi program pada pengujian overhead, saat eksekusi fungsi program hanya menggunakan CPU sekitar 0,24% pada "check_user.py". Dengan pengujian

perbandingan di temukan bahwa “check_user.py” hanya menggunakan CPU sekitar 0,1%. Dalam pengujian ini juga yang berlangsung selama 15 detik, “check_user.py” memiliki lebih banyak idle dari pada “check_user_na.py”.

Dengan pertimbangan di atas, penulis menyimpulkan bahwa data yang di peroleh pada “check_user.py” di pengaruhi oleh banyaknya kondisi idle saat eksekusi program di bandingkan dengan “check_user_na.py” yang dapat menyelesaikan program lebih cepat.

Dari data yang di peroleh dari pengujian ini, penulis dapat menyimpulkan bahwa tidak adanya jeda dalam mendeteksi kartu adalah yang menyebabkan penggunaan CPU yang tinggi pada idle, karena pada saat idle itulah perangkat terus menerus memeriksa apakah ada kartu yang valid di scanner. Dan data dapat di konfirmasi kebenarannya karena 2 program yang di pilih penulis menggunakan 2 bahasa pemograman yang berbeda, top dengan bahasa C dan psutil dengan Bahasa Python.

Sementara untuk perbedaan 2 program pada tugas akhir, “check_user.py” dan “check_user_na.py”, pada sisi RAM tidak terlalu banyak perbedaan. Namun pada sisi CPU walaupun data menunjukkan bahwa “check_user.py” memiliki rata – rata penggunaan lebih kecil meski menggunakan RFID dan TOTP, penulis lebih memihak kepada “check_user_na.py” yang memiliki rata – rata lebih tinggi namun dengan jeda waktu yang lebih singkat.

5. Kesimpulan

Masalah yang di angkat pada tugas akhir ini adalah adanya celah keamanan yang dapat di serang di *smart lock* pada komunikasi antara *smart lock* dengan pengguna. Solusi yang di tawarkan pada tugas akhir ini adalah perangkat *smart lock* yang bisa melakukan mengendalikan kunci dengan autentikasi user melalui 2 tahap autentikasi (RFID dan kode TOTP) dan menyimpan data RFID dan *secret* per RFID secara *offline*. Perangkat dapat melakukan proses autentikasi RFID dan autentikasi TOTP dengan baik dan hanya memerlukan waktu selama 3 detik dari proses awal pembukaan kunci hingga kunci terbuka.

Smart lock ini memiliki sistem keamanan yang cukup dasar, seperti tahan dari serangan Denial of Service pada RFID. Namun, RFID pada sistem masih bisa di serangan melalui rfid cloning. Hal ini membuat perangkat rentan terhadap serangan *Social Engineering* dimana penyerang bisa mengelabui pengguna yang memiliki RFID terdaftar untuk membiarkan penyerang membuat salinan RFID yang valid dan mencoba login ke sistem atau dengan serangan *skimming* dimana penyerang menggunakan perangkat tertentu untuk men-scan RFID valid dan mengambil data yang ada di RFID. Tentunya dengan adanya 2 *step authentication*, 2 serangan di atas bisa di atasi karena penyerang masih belum bisa menyelesaikan serangannya. Pada perangkat, data untuk *secret* pada setiap RFID di amankan dengan hanya mengizinkan 1 username untuk membuka database yang berisi nama, rfid uid dan *secret*. Jika penyerang berhasil mengakses Raspberry Pi secara langsung dan berhasil mencari cara untuk mengakses database, penyerang bisa mengambil semua *secret* yang tersimpan. 2 *vulnerability* ini hanya bisa di atasi dengan membuat ketentuan agar semua pengguna yang memiliki RFID yang terdaftar untuk menjaga kartunya dengan baik dan mengekripsi data *secret* di database. Dengan melakukan enkripsi *secret*, penyerang tidak dapat mengolah data yang di peroleh sehingga tidak bisa menyelesaikan eksploitnya. Dengan mengekripsi data *secretm* di perlukan perubahan dalam melakukan proses autentiaksi OTP, karena library PyOTP hanya membaca *secret* dengan tipe string dengan format base32.

Dengan melihat hasil pengujian *overhead*, Raspberry Pi 3 B+ dapat menjalankan program dan GPIO dapat mengendalikan semua sensor dan komponen dengan baik. Hasil ini tentunya tidak akan sama jika menggunakan *single-board computer* lainnya seperti LattePanda [19] yang di beroperasi dengan CPU Intel Cherry Trail Z8350 Quad Core di 1,92GHz yang menggunakan OS Windows. Bedanya spesifikasi ini tentunya akan mempengaruhi hasil yang di dapat dengan Raspberry Pi di bandingkan dengan yang di dapat dengan LattePanda.

Referensi

[1] Grand View Research, "Smart Lock Market Size, Share, Industry Research Report 2027," Februari 2020. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/smart-lock-market. [Accessed 28 12 2020].

[2] E. Rawes and J. Velasco, "The best smart locks for 2020," Digital Trends, 1 12 2020. [Online]. Available: https://www.digitaltrends.com/home/best-smart-locks/. [Accessed 30 12 2020].

[3] K. Marciniak, "Digital lockpicking - stealing keys to the kingdom," F-Secure Labs, 11 Desember 2019. [Online]. Available: https://labs.f-secure.com/blog/digital-lockpicking-stealing-keys-to-the-kingdom. [Accessed 30 12 2020].

[4] RASPBERRY PI FOUNDATION, "Remote Access," RASPBERRY PI FOUNDATION, [Online]. Available: https://www.raspberrypi.org/documentation/remote-access/. [Accessed 12 30 2020].

[5] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song and D. Wagner, "Smart Locks: Lessons for Securing Commodity Internet of Things Devices," in *ASIA CCS '16*, 2016.

[6] S. Kavde, R. Kavde, S. Bodare and G. Bhagat, "Smart digital door lock system using Bluetooth technology," in *2017 International Conference on Information Communication and Embedded Systems (ICICES)*, Chennai, 2017.

[7] M. E. Beqqal and M. Azizi, "Review on security issues in RFID systems," *Advances in Science, Technology and Engineering Systems Journal*, vol. II, no. 6, pp. 194-202, 2017.

[8] RASPBERRY PI FOUNDATION, "Raspberry Pi 3 Model B+," RASPBERRY PI FOUNDATION, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed 1 1 2021].

[9] D. M'Raihi, S. Machani, M. Pei and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," May 2011. [Online]. Available: https://tools.ietf.org/pdf/rfc6238.pdf. [Accessed 3 1 2021].

[10] PyOTP Contributors, "PyOTP - The Python One-Time Password Library," [Online]. Available: https://pyauth.github.io/pyotp/. [Accessed 5 January 2021].

[11] NXP Semiconductors, "MFRC522 Datasheet," 27 April 2016. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. [Accessed 26 1 2021].

[12] E. Young and M. Macdonald-Wallace, "MFRC522-python," [Online]. Available: https://github.com/pimylifeup/MFRC522-python. [Accessed 26 1 2021].

[13] Adafruit, "Adafruit Optical Fingerprint Sensor," 02 Februari 2021. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-optical-fingerprint-sensor.pdf. [Accessed 14 Februari 2021].

[14] K. Michael, "top(1) — Linux manual page," September 2020. [Online]. Available: https://man7.org/linux/man-pages/man1/top.1.html. [Accessed 24 1 2021].

[15] G. Rodola, "Psutil," [Online]. Available: https://psutil.readthedocs.io/en/latest/#. [Accessed 01 02 2021].

[16] D. Wang, Q. Gu, X. Huang and P. Wang, "Understanding Human-Chosen PINs:Characteristics, Distribution and Security," in *ACM ASIACCS 2017*, 2017.

[17] Common Vulnerabilities and Exposures, "CVE-2015-7225," 2015. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-7225. [Accessed 14 Februari 2020].

[18] "High CPU usage on idle state," 2 Mei 2019. [Online]. Available: https://github.com/pimylifeup/MFRC522-python/issues/9.

[19] LattePanda, "LattePanda 4G/64G," [Online]. Available: https://www.lattepanda.com/products/3.html. [Accessed 16 Februari 2021].