# 

**IMPLEMENTASI *TWO FACTOR AUTHENTICATION* DAN**

**ALGORITMA RSA SEBAGAI METODE AUTENTIKASI**

**LOGIN PADA SI-ABKA (SISTEM AMAL BAKTI**

**KEMENTERIAN AGAMA)**

**SKRIPSI**

oleh

**Ahmad Choirul Mustaqim**

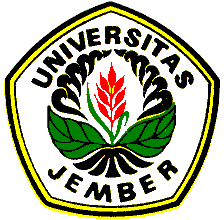
**NIM 152410101155**

**PROGRAM STUDI SISTEM INFORMASI**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS JEMBER**

**2019**



**IMPLEMENTASI *TWO FACTOR AUTHENTICATION* DAN**

**ALGORITMA RSA SEBAGAI METODE AUTENTIKASI**

**LOGIN PADA SI-ABKA (SISTEM AMAL BAKTI**

**KEMENTERIAN AGAMA)**

**SKRIPSI**

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan pendidikan sarjana (S1) Program Studi Sistem Informasi Universitas Jember dan mendapat gelar Sarjana Komputer

oleh

**Ahmad Choirul Mustaqim**

**NIM 152410101155**

**PROGRAM STUDI SISTEM INFORMASI**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS JEMBER**

**2019**

# PERSEMBAHAN

**MOTTO**

# PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Ahmad Choirul Mustaqim

NIM : 152410101155

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul “Implementasi *Two factor authentication* Dan Algoritma Rsa Sebagai Metode Autentikasi Login Pada Si-Abka (Sistem Amal Bakti Kementerian Agama)”adalah benar-benar hasil karya saya sendiri, kecuali jika ada pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada instansi manapun, dan bukti karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika dikemudian hari pernyataan ini tidak benar.

Jember, 20 Mei 2019

Yang menyatakan,

Ahmad Choirul Mustaqim

NIM 152410101155

**SKRIPSI**

**Implementasi *Two factor authentication* Dan Algoritma Rsa**

**Sebagai Metode Autentikasi Login Pada Si-Abka**

**(Sistem Amal Bakti Kementerian Agama)**

oleh

**Ahmad Choirul Mustaqim**

**NIM 152410101155**

Pembimbing

Pembimbing Utama : Achmad Maududie S.Kom,M.Sc.

Pembimbing Anggota : Diksy Media Firmansyah S.Kom, M.Kom.

**PENGESAHAN PEMBIMBING**

Skripsi berjudul “Implementasi *Two factor authentication* Dan Algoritma Rsa Sebagai Metode Autentikasi Login Pada Si-Abka (Sistem Amal Bakti Kementerian Agama)”, telah diuji dan disahkan pada:

Hari tanggal : Jumat, 10 Mei 2019

Tempat : Program Studi Sistem Informasi Universitas Jember

Disetujui oleh:

|  |  |
| --- | --- |
| Pembimbing I, | Pembimbing II, |

**PENGESAHAN PENGUJI**

Skripsi berjudul “Implementasi *Two factor authentication* Dan Algoritma Rsa Sebagai Metode Autentikasi Login Pada Si-Abka (Sistem Amal Bakti Kementerian Agama)”, telah diuji dan disahkan pada:

Hari tanggal : Jumat, 10 Mei 2019

Tempat : Program Studi Sistem Informasi Universitas Jember

Disetujui oleh:

|  |  |
| --- | --- |
| Pembimbing I, | Pembimbing II, |

Mengesahkan

Dekan Fakultas Ilmu Komputer,

Prof. Saiful Bukhori,ST., M.Kom

NIP. 196811131994121001

# RINGKASAN

# PRAKATA

# DAFTAR ISI

[PERSEMBAHAN ii](#_Toc25324264)

[PERNYATAAN iv](#_Toc25324265)

[PENGESAHAN PEMBIMBING vi](#_Toc25324266)

[PENGESAHAN PENGUJI vii](#_Toc25324267)

[RINGKASAN viii](#_Toc25324268)

[PRAKATA ix](#_Toc25324269)

[DAFTAR ISI x](#_Toc25324270)

[DAFTAR GAMBAR xii](#_Toc25324271)

[DAFTAR TABEL xiii](#_Toc25324272)

[BAB 1. PENDAHULUAN 1](#_Toc25324273)

[1.1 Latar belakang 1](#_Toc25324274)

[1.2 Rumusan masalah 3](#_Toc25324275)

[1.3 Batasan masalah 3](#_Toc25324276)

[1.4 Tujuan penelitian 4](#_Toc25324277)

[BAB 2. TINJAUAN PUSTAKA 5](#_Toc25324278)

[2.1 Password dan username 5](#_Toc25324281)

[2.2 Otentikasi 5](#_Toc25324282)

[2.3 SI-Abka 6](#_Toc25324283)

[2.4 *One time password* 6](#_Toc25324284)

[2.5 Time-Based OTP 6](#_Toc25324285)

[2.6 Algoritma RSA 7](#_Toc25324286)

[2.7 Penelitian Terdahulu 10](#_Toc25324287)

[BAB 3. METODOLOGI PENELITIAN 12](#_Toc25324288)

[3.1 Objek Penelitian 12](#_Toc25324289)

[3.2 Tempat Penelitian 12](#_Toc25324290)

[3.3 Tahapan Penelitian 12](#_Toc25324291)

[3.4 Pengembangan Software 13](#_Toc25324292)

[3.5 Pembuatan modul TOTP 13](#_Toc25324293)

[3.6 Implementasi RSA 14](#_Toc25324294)

[3.7 Uji keamanan 16](#_Toc25324295)

[3.7.1. Uji brute force 16](#_Toc25324305)

[3.7.2. Uji MITM 16](#_Toc25324306)

[BAB 4. HASIL DAN PEMBAHASAN 18](#_Toc25324307)

[4.1 Hasil Implementasi pembangkitan *secret* dan *public key* 18](#_Toc25324313)

[4.2 Hasil Implementasi proses login TOTP 22](#_Toc25324314)

[4.3 Hasil pembuatan model TOTP dan Algoritma rsa 25](#_Toc25324315)

[4.4 Hasil pengujian uji keamanan 32](#_Toc25324316)

[4.4.1. *Pengujian* *brute force* 32](#_Toc25324317)

[4.4.2. Pengujian MITM 36](#_Toc25324318)

[BAB 5. PENUTUP 39](#_Toc25324319)

[5.1 Kesimpulan 39](#_Toc25324326)

[5.2 Saran 41](#_Toc25324327)

[Daftar pustaka 42](#_Toc25324328)

[LAMPIRAN 44](#_Toc25324329)

# DAFTAR GAMBAR

[Gambar 2.1 Flowchart Proses Algoritma RSA 10](#_Toc25324331)

[Gambar 3.1 Alur Tahapan Penelitian 13](#_Toc25324332)

[Gambar 3.2. Flowchart Impelentasi RSA 15](#_Toc25324333)

[Gambar 3.3. cara kerja MITM 17](#_Toc25324334)

[Gambar 4.1. flowchart implementasi public key dan secret key 19](#_Toc25324335)

[Gambar 4.2. flowchart implementasi proses login TOTP 22](#_Toc25324336)

[Gambar 4.3. tampilan login SI-Abka 26](#_Toc25324337)

[Gambar 4.4. input kode TOTP 26](#_Toc25324338)

[Gambar 4.5. input kode TOTP dengan qr-code 27](#_Toc25324339)

[Gambar 4.6. aplikasi ChoiTOTP 27](#_Toc25324340)

[Gambar 4.7. menu aplikasi ChoiTOTP 28](#_Toc25324341)

[Gambar 4.8. tampilan scan qr-code 29](#_Toc25324342)

[Gambar 4.9.kode TOTP berhasil didapatkan 29](#_Toc25324343)

[Gambar 4.10. memasukan kode TOTP 30](#_Toc25324344)

[Gambar 4.11.dasboard user 30](#_Toc25324345)

[Gambar 4.12. dashboard admin 31](#_Toc25324346)

[Gambar 4.13. pemberitahuan kode OTP salah 31](#_Toc25324347)

[Gambar 4.14. penentuan target sistem 33](#_Toc25324348)

[Gambar 4.15. variabel header 34](#_Toc25324349)

[Gambar 4.16. variabel payload brute force 34](#_Toc25324350)

[Gambar 4.17. hasil brute force 35](#_Toc25324351)

[Gambar 4.18. respons code OTP benar 35](#_Toc25324352)

[Gambar 4.19. proses scan ip target 36](#_Toc25324353)

[Gambar 4.20. arp poisoning ip 37](#_Toc25324354)

[Gambar 4.21. pembelokan data komunikasi dengan router 37](#_Toc25324355)

[Gambar 4.22. dekripsi ssl 38](#_Toc25324356)

[Gambar 4.23. hasil pembacaan packet data 38](#_Toc25324357)

# DAFTAR TABEL

# PENDAHULUAN

Bab ini menjelaskan hal-hal yang berkaitan dengan pendahuluan penelitian. Adapun pembahasan pada bab ini meliputi latar belakang, rumusan masalah, tujuan dan manfaat, serta batasan masalah.

## Latar belakang

Di era teknologi internet sekarang ini, semua informasi dapat dikirim dengan bebas melalui suatu jaringan dengan tingkat keamanan yang rentan dan memungkinkan terjadinya penyadapan suatu informasi. Hal tersebut secara langsung maupun tidak langsung mempengaruhi sistem perdagangan, transaksi, bisnis, perbankan, industri dan pemerintahan yang umumnya mengandung informasi rahasia. Keamanan data saat ini sangat penting mulai dari mengamankan data yang disimpan sampai data yang dikirim. Data yang bersifat rahasia perlu dibuatkan suatu sistem penyimpanan dan pemrosesan khusus agar data tersebut tidak mudah di baca atau diubah oleh pihak yang tidak berwenang.

Berbagai hal telah dilakukan untuk mendapatkan jaminan keamanan informasi rahasia. Faktor utama yang harus dipenuhi dalam mengamankan data rahasia adalah tingkat keamanan teknologi informasi yang tinggi. Data tidak hanya berupa data atau teks, *login* ke dalam suatu sistem perlu di enkripsi agar hanya orang yang memiliki akses yang dapat masuk ke dalam sistem.

Proses autentikasi pada prinsipnya berfungsi sebagai kesempatan pengguna dan pemberi layanan dalam proses pengaksesan *resource*. Pengguna harus mampu memberikan informasi yang dibutuhkan pemberi layanan untuk berhak mendapatkan *resource*nya. Sedangkan pihak pemberi layanan harus mampu menjamin bahwa pihak yang tidak berhak tidak akan dapat mengakses *resource* ini (Khairina, 2011). Jika seseorang sudah mengetahui *password* kita, maka akun tersebut mudah sekali disalah gunakan tanpa sepengetahuan pemilik aslinya.

Salah satu cara yang digunakan adalah dengan menyandikan isi informasi menjadi suatu kode-kode yang tidak dimengerti sehingga penyadap akan kesulitan

untuk mengetahui isi informasi yang sebenarnya. Dari masalah tersebut perlu adanya suatu metode *login* yang dapat mengamankan akun dari adanya percobaan pembobolan. Salah satu sistem yang memerlukan pengamanan ekstra antara lain sistem perbankan, karena perbankan menyimpan banyak data nasabah dan data transaksi sampai data keuangan yang rentan terhadap perubahan sekecil apapun.

SI-Abka (sistem amal bakti kementerian agama jember) merupakan sistem yang mengelola data koperasi dari seluruh anggota yang bekerja di bawah instansi kementerian agama jember. Sistem ini berfungsi sebagai pengelola data mulai dari data anggota, data simpanan, sampai data pinjaman. Data-data tersebut sangat rentan terhadap perubahan karena menyangkut keuangan nasabah dan koperasi. Saat ini SI-Abka hanya menggunakan *username* dan *password* untuk metode autentikasi nya. Penggunaan *username* dan *password* rentan terhadap pembobolan, sehingga perlu adanya teknologi tambahan untuk meningkatkan keamanan saat melakukan otentikasi ke sistem. Teknologi yang dibutuhkan yaitu OTP (*one time password*).

Proses login yang sebelumnnya hanya mengandalkan username dan *password* akan ditambah dengan memasukan kode OTP. Proses tersebut dinamakan *two factor authentication*, yaitu menggunakan kode OTP sebagai pengaman tambahan (Sudiarto Raharjo, E.K. Ratri, dan Susilo 2017). Kode OTP ini otomatis dibangkitkan sesuai dengan waktu dan parameter tertentu dan dapat di akses dengan menggunakan aplikasi android, atau hardware khusus. Kelebihan OTP berbasis waktu adalah tidak mengandalkan server saat pembangkitan kode OTP sehingga meminimalisir adanya kode OTP yang lama tersampaikan dan tidak perlu adanya penyimpanan kode OTP ke dalam database. Urutan angka yang muncul dalam OTP One Time Password ini diatur berdasarkan sinkronisasi dengan waktu dan urutan angka OTP unik untuk setiap akun hanya diketahui oleh token OTP dan server OTP penyedia layanan .

Proses pembangkitan kode OTP juga menggunakan algoritma RSA sehingga hasil pembangkitan kode OTP sangat susah di prediksi dan bersifat sangat random. Algoritma RSA juga berjalan di dua sisi yaitu di sisi server dan sisi client yang berupa android. Kode OTP berbasis waktu memiliki pola tersendiri dan jika terdapat orang yang berniat jahat dan mengetahui pola tersebut maka rawan akan terjadinya pembobolan. Oleh karena itu terdapat algoritma RSA yang akan mengenkripsi kode OTP sehingga data yang dikirim bukan kodenya secara langsung.

Kasus pembobolan 13 juta akun bukalapak pada tahun 2018 silam merupakan contoh rawannya penggunaan username dan password. Data akun yang sudah diketahui publik akan dengan mudah dimanfaatkan dan dapat merugikan pemilik aslinya. Dengan menambahkan *two factor authentication*  dan algoritma rsa diharapkan dapat mengamankan data akun yang disimpan maupun yang telah dicuri.

## Rumusan masalah

Berdasarkan latar belakang masalah penelitian, maka muncul perumusan masalah sebagai berikut.

1. Bagaimana meningkatkan keamanan pada proses otentikasi SI-ABKA?
2. Mengapa perlu meningkatkan keamanan *Two factor authentication* menggunakan algoritma RSA ?
3. Bagaimana proses pembangkitan kode OTP di sistem SI-ABKA?
4. Bagaimana proses pengamanan kode OTP dengan menggunakan algoritma RSA?
5. Berapa tingkat keamanan sistem dengan *Two factor authentication* dan algoritma RSA pada otentikasi SI-ABKA?

## Batasan masalah

Peneliti memberikan batasan masalah untuk objek dan tema yang dibahas sehingga tidak terjadi penyimpangan dalam proses penelitian dan menganalisis

* + 1. Bahasa pemrograman yang digunakan adalah PHP untuk sistem SI-Abka dan java android untuk membangkitkan kode OTP*.*
    2. Algoritma kriptografi yang di gunakan adalah RSA.
    3. Nasabah dapat mengoperasikan android.
    4. Tahapan testing menggunakan teknik *brute force* dan *man in the midle* (MITM) terhadap token TOTP.
    5. Kode OTP memiliki masa aktif selama 30 detik.

## Tujuan penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah:

* + 1. Mengimplementasikan *Two factor authentication* ke dalam sistem SI-ABKA
    2. Meningkatkan keamanan transaksi pada SI-ABKA dengan menggunakan *two factor authentication.*
    3. Membangkitkan kode OTP dengan menggunakan *moving factor* berupa variabel waktu
    4. Mengimplementasikan algoritma RSA untuk pengamanan kode OTP
    5. Mengukur tingkat keamanan otentikasi SI-ABKA dengan menggunakan *security test brute force* dan MITM

# TINJAUAN PUSTAKA

Pada bagian ini dipaparkan tinjauan yang berkaitan dengan masalah yang dibahas, serta kajian teori yang dikaitkan dengan permasalahan yang dihadapi. Teori yang di dapatkan berupa pembangkitan OTP dan penerapanya yang dapat membantu peneliti dalam penelitian ini. Selain pembangkitan OTP penulis juga memelajari algoritma RSA. Perhitungan yang di dapatkan akan membantu peneliti dalam menghitung kode OTP yang akan di *generate* secara berkala oleh server dan *client.* Hasil perhitungan akan di proses oleh *client* dan server yang akan digunakan untuk proses autentikasi OTP di sistem SI-ABKA.



## Password dan username

Password atau kata sandi dapat digunakan untuk layanan otentikasi, yaitu layanan yang berhubungan identifikasi, baik mengidentifikasi kebenaran pihak – pihak yang berkomunikasi ( *user authentication* atau *entity authentication* ) maupun mengidentifikasi kebenaran sumber pesan. “Otentikasi sumber pesan secara benar memberikan kepastian integritas data” (Inayatullah, 2007). *Password* bersifat statis atau sama, maksud statis disini adalah nilai atau *values* dari *password* tersebut sama dengan *password* sebelumnya hingga user menggantinya. Biasanya user mengganti *password* ketika sudah merasa bahwa akun dia sudah tidak aman atau sudah diketahui oleh orang lain.

## Otentikasi

Otentikasi (*Authentication*) adalah proses untuk memastikan bahwa kedua ujung koneksi dalam keadaan benar atau sama (Mustofa, 2003). Seperti password pada umumnya, syarat agar otentikasi berhasil adalah password yang dikirimkan client harus sama dengan *password* yang disimpan di server. Dengan alasan keamanan jarang sekali server menyimpan *password user* dalam bentuk *plain text*. Biasanya server menyimpan password user dalam bentuk hash sehingga tidak bisa dikembalikan dalam bentuk *plain text.* Jadi syarat otentikasi berhasil di atas bisa diartikan sebagai hasil penghitungan hash dari password yang dikirim klien harus sama dengan nilai *hash* yang disimpan dalam server.

## SI-Abka

Sistem informasi amal bakti kementerian agama (SI-abka) merupakan sistem web yang membantu koperasi amal bakti kementerian agama jember dalam melakukan transaksi. Sistem ini menangani data dan informasi anggota, transaksi simpan pinjam, sampai menangani rekap pembayaran cicilan oleh bendahara di tiap satuan kerja.

## *One time password*

Dikutip dari (Musliyana, Arif, & Munadi, 2016) bahwa *One time password* (OTP) merupakan metode otentikasi yang menggunakan *password* yang selalu berubah setelah setiap kali *login*, atau berubah setiap interval waktu tertentu. *One time password* ini haruslah *password* yang acak sehingga sulit ditebak oleh orang lain. Keuntungan dari *one time password* adalah pencegahan penyalahgunaan username dan password yang biasanya statis. Dengan tambahan *one time password* ini maka login tidak bisa ditiru oleh orang lain. Keuntungan ini berarti jika berhasil seorang mendapatkan username dan password, maka tidak dapat digunakan karena dia harus memasukkan *one time password* yang lain.

## Time-Based OTP

OTP jenis ini berbasis sinkronisasi waktu yang berubah secara konstan pada setiap satuan interval waktu tertentu. Proses ini memerlukan sinkronisasi antara token milik client dengan server otentikasi. Pada jenis token yang terpisah (*disconnected token*), sinkronisasi waktu dilakukan sebelum token diberikan kepada client (Kim, Lee, Lee, & Jun, 2009). Tipe token lainnya melakukan sinkronisasi saat token dimasukkan dalam suatu alat input.

Setiap token memiliki sebuah jam akurat yang telah disinkronisasikan dengan waktu yang terdapat pada server otentikasi. Pada sistem OTP ini, waktu merupakan bagian yang penting dari algoritma password, karena pembangkitan password baru didasarkan pada waktu dan kunci rahasia saat itu dan bukan pada password sebelumnya .

Pada OTP jenis ini sudah mulai diimplementasikan terutama pada remote *Virtual Private Network* (VPN), dan keamanan jaringan Wi-Fi dan juga pada berbagai aplikasi *Electronic Commerce* (E-commerce). Ukuran standar penggunaan waktu pada algoritma ini adalah 30 detik (M’Raihi, Machani, Pei, & Rydell, 2011). Nilai ini dipilih sebagai keseimbangan antara keamanan dan kegunaan. Pada penelitian ini, OTP yang digunakan berbasis sinkronisasi waktu dengan kombinasi Algoritma RSA.

## Algoritma RSA

Rivest Shamir Adleman (RSA) adalah salah satu algoritma kriptografi asimetris (kriptografi kunci - publik) yaitu menggunakan dua kunci yang berbeda ( private key dan *public key*). Kekuatan algoritma RSA tidak hanya terletak pada panjang kuncinya (semakin panjang kunci, maka semakin lama waktu kerja) dan penggunaan kunci - publik dan kunci privat pada umumnya (Muchlis, Budiman, & Rachmawati, 2007). Algoritma ini membantu dalam pembangkitan kode OTP agar lebih aman dan tidak mudah di tebak. Pembangkitan OTP dibangun berdasarkan algoritma tersendiri jika algoritma tersebut diketahui maka kode OTP sangat mudah di ketahui, oleh karena itu dibutuhkan algoritma kriptografi agar hasil OTP lebih aman.

Terdapat 3 proses yang dilakukan dalam proses algoritma ini yaitu:

1. Pembentukan kunci

Proses pembangkitan kunci pada kriptografi RSA adalah sebagai berikut :

* 1. Pilih dua buah bilangan prima sembarang *p* dan *q*. Misal p =13 dan q =17 jaga kerahasiaan *p* dan *q*.
  2. Hitung nilai untuk kunci public n dengan formula . Nilai n digunakan untuk kunci public.
  3. Hitung . Setelah telah dihitung, *p* dan *q* dapat dihapus untuk mencegah diketahuinya oleh pihak lain.
  4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya *e,* yang relatif prima terhadap *m* (relatif prima berarti GCD(e, m) = 1) dengan syarat , , dan . Misal e = 5;
  5. Cari kunci d dengan menggunakan algoritma eucleadean yang di perluas, yaitu dengan cara berikut :

🡺

🡺

🡺

t0 = 0;

t1 = 1;

t2 = t0 – q1.t1 = 0 – 38(1) = -38

t3 = t1 – q2.t2 = 1 – 2(-38) = 77

didapatkan kunci d = 77

* 1. didapatkan kunci

n (public) = 221

e (enkripsi) = 5

d (dekripsi) = 77

1. Proses Enkripsi

Setelah proses proses pembangkitan kunci selesai, kemudian lanjut ke proses enkripsi pesan menggunakan kunci publik dari hasil pembangkitan kunci dengan menggunakan rumus (1) berikut:

Menggunakan kunci yang diperoleh di atas kita akan mencoba untuk malakukan enkripsi pesan sederhana. Misalnya pesan yang akan di enkripsi adalah angka 48 dengan nama P (plain), maka akan diperoleh C (chiper) dengan perhitungan sebagai berikut:

Jadi hasil enkripsi 48 menggunakan kunci yang diperoleh di atas adalah 29.

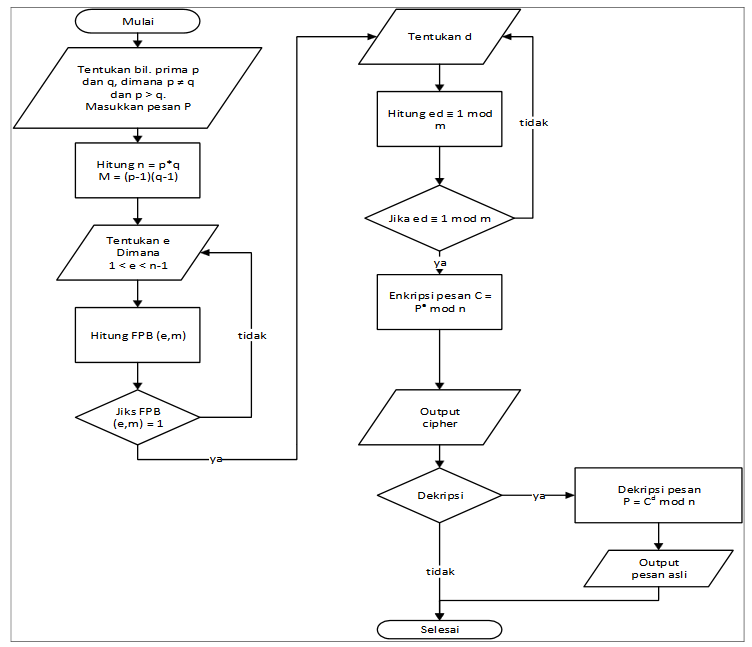
1. Proses Dekripsi

Untuk mengembalikan pesan *ciphertext* menjadi *plaintext* (pesan asli) adalah dengan menggunakan rumus dekripsi RSA sebagai berikut:

*P =*

Dengan menggunakan pesan hasil enkripsi dan kunci yang diperoleh di atas dapat dilakukan dekripsi pesan seperti pada bentuk (4) berikut:

Dari hasil dekripsi di atas dapat dibuktikan bahwa hasil enkripsi pesan dapat didekripsi kembali ke pesan asli.



Gambar 2.1 Flowchart Proses Algoritma RSA

## Penelitian Terdahulu

Penelitian dengan judul “Implementasi Algoritma Time-Based *One time password* Dalam Otentikasi Token Internet Banking” (Ungkawa, Dewi, & Putra, t.t.). Penelitian ini melakukan penerapan TOTP dalam pembangkitan token OTP nya. Token tersebut tidak langsung dikirim ke user tetapi mengirim nilai hash nya. Penelitian ini menggunakan hash SHA256 sebagai metode hashingnya dan emkripsi AES. Penelitian ini diaplikasikan pada sistem internet banking di mana antara token virtual dan server dipasang algoritma TOTP untuk menghasilkan password sebagai otentikasi tambahan . Dari hasil pengujian yang dilakukan bahwa password OTP tidak muncul secara berulang dan *secret key* yang dihasilkan secara acak juga tidak muncul secara berulang tetapi mempunyai presentase kemiripan tertinggi sebesar 0,03%.

Penelitian dengan judul “Aplikasi Algoritma RSA untuk Keamanan Data pada Sistem Informasi Berbasis Web” (Rosnawan, 2011). Untuk menjaga keamanan dari password dan pesan berupa file, biasanya digunakan teknik enkripsi agar kerahasiaan data terebut terjamin. Salah satu algoritma enkripsi yang sering digunakan adalah algoritma RSA. Pada kesempatan ini penulis tertarik mengkaji tentang aplikasi pengamanan data pada sistem informasi berbasis web. Permasalahan dalam skripsi ini adalah bagaimana implementasi algoritma RSA untuk keamanan data pada sistem informasi berbasis web.

Penelitian dengan judul “Implementasi Algoritma RSA Untuk Enkripsi Dan Dekripsi Sms (*Short Message Service*) Pada Ponsel Berbasis Android” (Sardju, Magdalena, & Atmaja, 2015). Penelitian ini membahas tentang keamanan dalam penggunaan servis sms. Peneliti mengamankan pesan sms dengan menggunakan algoritma RSA. Hasil keluaran dari sistem ini yaitu pada pengiriman sms yang telah terenkripsi akan terkirim apabila ≤ 160 karakter, dan sms tidak akan terkirim apabila ≥ 160 karakter, pada proses enkripsi dan dekripsi membutuhkan waktu rata-rata 0,18 detik, pada pengujian *avalanche effect* dengan menggunakan masukan plaintext yang berbeda tiap percobaan akan menghasilkan chipert ext yang berbeda dengan presentase rata-rata sebesar 10,35 %, sedangkan pada pengujian *brute force* membutuhkan waktu selama 1,652 x tahun untuk mencoba semua kemungkinan kunci yang ada.

Penelitian - penelitian di atas dapat disimpulkan bahwa *two factor autentication* dan algoritma RSA sesuai untuk mengamankan fungsi login di sistem SI-Abka. Diharapkan dengan penelitian ini keamanan transaksi di sistem tersebut lebih tinggi lagi dan aman.

# METODOLOGI PENELITIAN

Pada bab ini akan membahas objek penelitian, tempat penelitian, tahapan penelitian, dan studi literatur yang digunakan dalam pembangunan modul sistem login pada sistem SI-abka dan implementasi algoritma RSA aplikasi tersebut untuk menjaga keaslian kerahasiaan transaksi data antara client dan server.

## Objek Penelitian

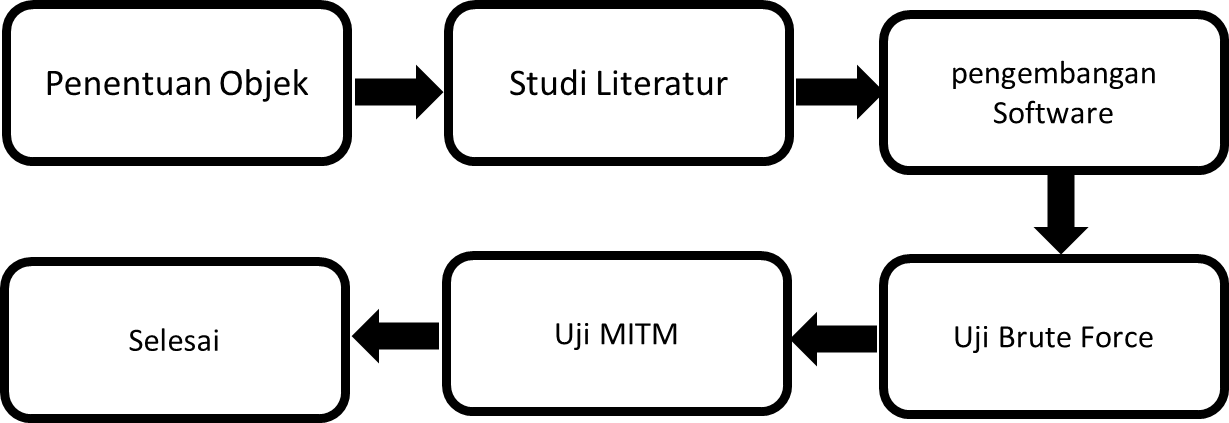
Objek penelitian merupakan sistem SI-Abka yang ada pada koperasi kementeriam agama jember. Aplikasi tersebut penggunakan web php dan database mysql. Proses autentikasi dari seluruh akun hanya menggunakan username dan password. Dari sistem tersebut akan ditambah *two factor authentication* berupa TOTP dan algoritma RSA. Sistem authenticaton tambahan tersebut diharapkan dapat memperkuat keamanan sistem SI-Abka. Kode OTP akan di generate atau dibangkitkan menggunakan aplikasi di HP android atau sebuah alat portable. Kode di bangkitkan dengan cara memasukan *public key* ke dalam sistem SI-Abka saat meregistrasikan aplikasi android dengan sistem agar dapat selarah.

## Tempat Penelitian

Tempat dilaksanakan penelitian yaitu di Kementerian Agama Kabupaten Jember. SI-Abka di terapkan pada Koperasi Amal Bakti Kementerian Agama sebagai sistem yang membantu pelayanan di koperasi.

## Tahapan Penelitian

Tahapan Penelitian yang akan dilakukan dapat dilihat dalam diagram alur di bawah ini. Penelitian ini terdiri dari 8 tahap mulai dari perencanaan, implementasi sampai tahap testing. Tahap-tahap ini harus di lakukan secara urut karena tahap sebelumnya berpengaruh ke tahap selanjutnya.



Gambar 3.1 Alur Tahapan Penelitian

## Pengembangan Software

Sistem SI-abka merupakan sistem yang sedang di kembangkan dan masih memerlukan beberapa fitur tambahan untuk membantu pengelolaan administrasinya.

Bab ini merupakan bagian yang membahas tentang pengembangan sistem *two factor authentication* dan Algoritma rsa sebagai metode autentikasi Login. Pengembangan sistem dilakukan dengan menggunakan model waterfall, dengan tahapan yakni analisis kebutuhan fungsional dan non-fungsional sistem, pembuatan desain sistem, penulisan kode program dan pengujian sistem.

## Pembuatan modul TOTP

Pada tahap implementasi ini, dilakukan dengan cara mentransformasikan desain sistem yang telah dibuat ke dalam sebuah bahasa pemrograman berorientasi objek sehingga dapat dihasilkan suatu modul login TOTP. Algoritma RSA digunakan untuk mengkripsi data yang dikirim dari client ke server. Pengguna dapat menggunakan beberapa aplikasi yang sudah ada di playstore atau aplikasi yang sudah dibuat oleh peneliti dalam pembangkitan kode OTP. Salah satu contoh aplikasi yang sudah ada antara lain google authenticator dan authy.

Saat pengguna akan meninputkan kode OTP maka aplikasi android akan membangkitkan TOTP berdasarkan data yang sama dengan server meskipun tidak berkomunikasi secara langsung. Setelah kode OTP berhasil dibangkitkan maka aplikasi android akan menengripsi kode tersebut menggunakan publik key yang didapat sebelumnya. Hasil enkripsi akan digunakan oleh pengguna dan dimasukan ke dalam sistem SI-ABKA. Saat sistem menerima inputan kode OTP maka akan di dekripsi dan dicocokan dengan hasil pembangkitan yang dilakukan oleh sistem. Jika sesuai maka perintah akan di loloskan jika tidak maka akan invalid dan gagal.

## Implementasi RSA

Algoritma RSA di gunakan untuk mengamankan data yang dikirim oleh client ke server. Dalam penelitian kali ini data yang dikirim hanya data OTP sebagai pembanding antara data yang dienkripsi dan yang tanpa enkripsi. Dengan penambahan enkripsi RSA diharapkan data tidak mudah di baca dan di tebak. Alur penggunaan algoritma RSA dalam penelitian ini dapat di lihat dalam flowchart berikut



Gambar 3.2. Flowchart Impelentasi RSA

## Uji keamanan

Uji keamanan dilakukan untuk mencoba keamanan sistem saat password dan username telah diketahui. Saat password dan username digunakan untuk login, sistem akan menampilkan form input kode OTP. Uji keamanan ini berfungsi untuk mengetahui seberapa besar dampak penggunaan *two factor authentication* dan algoritma RSA terhadap pengamanan proses login.



### Uji brute force

Algoritma *brute force* adalah algoritma yang memecahkan masalah dengan sangat sederhana, langsung, dan dengan cara yang jelas/lempang. Penyelesaian permasalahan password crackingdengan menggunakan algoritma *brute force* akan menempatkan dan mencari semua kemungkinan password dengan masukan karakter dan panjang password tertentutentunya dengan banyak sekali kombinasi password (Pramudita, 2010).

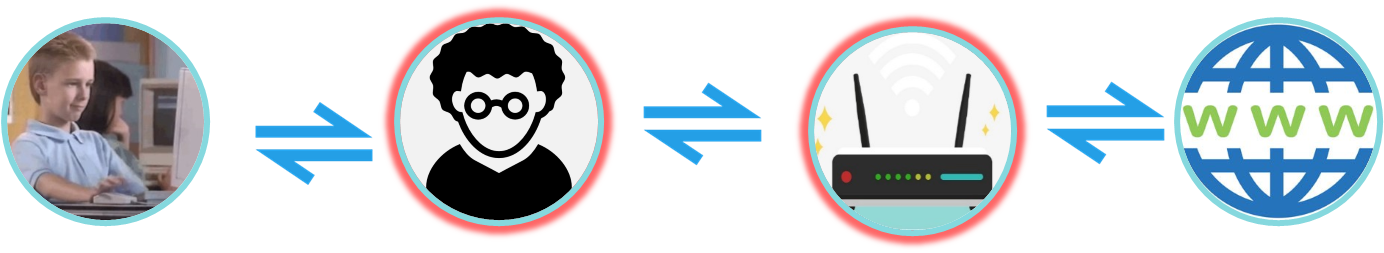
*Brute force* merupakan algoritma sederhana dalam proses pembuatan kemugkinan kode. Pengguna hanya tinggal memasukan panjang karakter dan ukuran kode yang akan dicari. Tiap kemungkinan kode akan di generate secara berurutan. Uji keamana kan di lakukan dengan cara memasukan kode OTP secara random dan cepat menggunakan metode *brute force* dengan bantuian aplikasi burp suite.

Cara kerja aplikasi tersebut adalah memotong jalur komunikasi dan melihat data yang dikirmkan. Data tersebut berisi data akun yang berupa data KTA,Password, dan kode OTP. Dalam pengujian kali ini kita akan menguji seberapa kuat dan berapa persen kemungkinan kode OTP akan diketahui.

### Uji MITM

Selain menggunakan teknik *brute force* pengujian juga menggunakan teknik *man in the middle*. Cara kerja teknik ini adalah mendengarkan/melihat *traffic* yang mengarah ke suatu situs web dan membaca setiap data yang dikirimkan, saat terdapat penguna yang login ke SI-Abka maka data yang di kirimkan dapat terbaca.

Cara kerja teknik ini adalah mendengarkan data yang dikirim ke server dengan mengelabui komputer client. Pada umumnya client berkomunikasi dengan server melalui sebuah perangkat router, tetapi jika terdapat serangan MITM diantara client dan router terdapat perangkat tambahan. Perangkat tambahan tersebut mengelabui komputer client seakan-akan bertindak sebagai router, seperti pada gambar 4.



Gambar 3.3. cara kerja MITM

Simulasi MITM menggunakan dua buah alat, salah satu sebagai target dan yang lain sebagai penyerang. Device target dapat menggunakan seluruh device windows, android, dll, sedangkan penyerang menggunakan OS kali linux. Teknik MITM menggunakan beberapa tools antara lain:

1. Netdiscover sebagai tool ip hunter (untuk mencari ip target)
2. Ettercap sebagai pembaca trafict data
3. Sslstrip sebagai pembelok website https
4. Wireshark untuk memfilter data yang telah dibaca ettercap

Data yang didapat dari teknik MITM berupa data post yang dikirim oleh client ke server. Data tersebut berupa text sehingga dapat dengan mudah dibaca tanpa bantuan tool khusus.

# HASIL DAN PEMBAHASAN



## Hasil Implementasi pembangkitan *secret* dan *public key*

Sistem SI-abka memiliki 3 level user dan seluruh user memiliki metode login yang sama yaitu menggunakan username dan password. Pada saat login pertama data user tersebut akan di lihat apakah sudah mendaftarkan akun tersebut dengan modul TOTP, jika sudah maka user tersebut akan langsung di arahkan ke form input kode OTP, jika tidak maka akan di arahkan ke tampilan generate kode OTP.

Saat terdapat user yang diarahkan ke page generate kode maka sistem akan menggenerate atau membuatkan suatu kode unik yaitu *secret key* kode RSA.Key yang dibuat berupa qrcode yang harus discan atau di pindai dengan aplikasi tertentu. Setelah di pindai maka alikasi tersebut akan menghitung dan menampilkan kode OTP sesuai dengan secret key tersebut. Setelah *secret key* telah dibuat maka sistem akan menunggu input kode TOTP oleh user. Jika kode tersebut benar maka *secret key* akan disimpan , jika tidak maka akan di arahkan ke tampilan login kembali.



Gambar 4.1. flowchart implementasi public key dan secret key

Dari flowchart tersebut menghasilkan file library yang dapat digunakan unruk membangkitkan secret key dan public key. Secret key dan public key tersbut dapat digunakan untuk melakukan enkripsi dan dekripsi dengan menggunakan algoritma RSA. Berikut source code library RSA yang berhasil dibuat

1. <?php
2. **class** libchoi {
3. **private** $p;
4. **private** $q;
5. **protected** $n;
6. **protected** $e;
7. **protected** $d;
8. **function** generate() {
9. **do** {
10. $rand1 = rand(100, 2000);
11. $rand2 = rand(100, 2000);
12. $this->p = gmp\_nextprime($rand1);
13. $this->q = gmp\_nextprime($rand2);
14. } **while** ($rand1 == $rand2);
15. $this->n = (int)$this->p \* $this->q;
16. $totient = gmp\_mul(gmp\_sub($this->p, 1), gmp\_sub($this->q, 1));
17. **for** ($cari\_e = 2;$cari\_e < $totient;$cari\_e++) {
18. $gcd = gmp\_gcd($cari\_e, $totient);
19. **if** (gmp\_strval($gcd) == '1') {
20. $this->e = $cari\_e;
21. **break**;
22. }
23. }
24. $i = 1;
25. **do** {
26. $res = gmp\_div\_qr(gmp\_add(gmp\_mul($totient, $i), 1), $this->e);
27. $i++;
28. **if** ($i == 10000) **break**;
29. }
30. **while** (gmp\_strval($res[1]) != '0');
31. $this->d = $res[0];
32. }
33. **function** enkrip($data) {
34. $angkaarray = str\_split($data);
35. $hasilenkrip = '';
36. **for** ($i = 0;$i < count($angkaarray);++$i) {
37. //rumus enkripsi <enkripsi>=<pesan>^<e>mod<n>
38. $hasilenkrip.= gmp\_strval(gmp\_mod(gmp\_pow(($angkaarray[$i]), $this->e), $$this->n));
39. **if** ($i != count($angkaarray) - 1) {
40. $hasilenkrip.= ".";
41. }
42. }
43. **return** $hasilenkrip;
44. }
45. **function** enkrip\_withkey($data, $user\_d, $user\_e) {
46. $angkaarray = str\_split($data);
47. $hasilenkrip = '';
48. **for** ($i = 0;$i < count($angkaarray);++$i) {
49. //rumus enkripsi <enkripsi>=<pesan>^<e>mod<n>
50. $hasilenkrip.= gmp\_strval(gmp\_mod(gmp\_pow(($angkaarray[$i]), $this->e), $$this->n));
51. **if** ($i != count($angkaarray) - 1) {
52. $hasilenkrip.= ".";
53. }
54. }
55. **return** $hasilenkrip;
56. }
57. **function** dekrip($data) {
58. $hasildekrip = "";
59. $teks = explode(".", $data);
60. **foreach** ($teks **as** $nilai) {
61. //rumus enkripsi <pesan>=<enkripsi>^<d>mod<n>
62. $hasildekrip.= (gmp\_strval(gmp\_mod(gmp\_pow(intval($nilai), intval($this->d)), $this->n)));
63. }
64. **return** $hasildekrip;
65. }
66. **function** dekrip\_withkey($data, $user\_n, $user\_d) {
67. $teks = explode(".", $data);
68. **foreach** ($teks **as** $nilai) {
69. //rumus enkripsi <pesan>=<enkripsi>^<d>mod<n>
70. $hasildekrip.= (gmp\_strval(gmp\_mod(gmp\_pow($nilai, $user\_d), $user\_n)));
71. }
72. **return** $hasildekrip;
73. }
74. **function** set\_Publickey($data) {
75. $pecahkey = explode(".", $data);
76. $this->n = $pecahkey[0];
77. $this->e = $pecahkey[1];
78. }
79. **function** set\_Privatekey($data) {
80. $pecahkey = explode(".", $data);
81. $this->n = $pecahkey[0];
82. $this->d = $pecahkey[1];
83. }
84. **function** get\_n() {
85. **return** $this->n;
86. }
87. **function** get\_e() {
88. **return** $this->n;
89. }
90. **function** get\_d() {
91. **return** $this->n;
92. }
93. **function** getPublicKey() {
94. $public = $this->n . "." . $this->e;
95. **return** $public;
96. }
97. **function** getPrivateKey() {
98. $private = $this->n . "." . $this->d;
99. **return** $private;
100. }
101. **function** getKey() {
102. $key = $this->n . "." . $this->e . "." . $this->d;
103. **return** $key;
104. }
105. ?>

Proses pembangkitan secret key dan public key algoritma RSA dijalankan saat pengguna melakukan proses login. Jika pengguna tidak memiliki secret key dan public key di database maka sistem akan membangkitkan key baru. Saat user memasukan kode OTP yang telah di enkripsi dan benar maka key tersebut akan disimpan. Jika tidak maka sistem akan mengembalikan ke menu login dan membuat pasangan secret key dan public key baru. Secret key dan public key tiap orang berbeda-beda.

## Hasil Implementasi proses login TOTP



Gambar 4.2. flowchart implementasi proses login TOTP

Dari flowchart tersebut menghasilkna source code yang berfungsi untuk membangkitkan secret key OTP yang dapat di scan oleh device pengguna dan sebagai fungsi untuk cek kode OTP apakah sudah sesuai atau belum. Berikut source code yang dihasilkan

1. <?php
2. **class** libchoi {
3. **protected** $length = 6; //panjang hasil key (default nya 6)
4. **public** **function** generateRandomSecret($secretLength = 32) //panjang secret key nya
5. {
6. $secret = '';
7. $validChars = **array**(
8. 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
9. 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
10. 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
11. 'Y', 'Z', '2', '3', '4', '5', '6', '7',
12. '=',
13. );
14. // Valid secret lengths are 80 to 640 bits
15. **if** ($secretLength < 16 || $secretLength > 128) {
16. **throw** **new** Exception('Bad secret length');
17. }
18. $random = false;
19. **if** (function\_exists('random\_bytes')) {
20. $random = random\_bytes($secretLength);
21. } **elseif** (function\_exists('mcrypt\_create\_iv')) {
22. $random = mcrypt\_create\_iv($secretLength, MCRYPT\_DEV\_URANDOM);
23. } **elseif** (function\_exists('openssl\_random\_pseudo\_bytes')) {
24. $random = openssl\_random\_pseudo\_bytes($secretLength, $cryptoStrong);
25. **if** (!$cryptoStrong) {
26. $random = false;
27. }
28. }
29. **if** ($random !== false) {
30. **for** ($i = 0;$i < $secretLength;++$i) {
31. $secret.= $validChars[ord($random[$i]) & 31];
32. }
33. } **else** {
34. **throw** **new** Exception('gagal buat random');
35. }
36. **return** $secret;
37. }
38. **public** **function** getCode($secret, $timeSlice = null) {
39. **if** ($timeSlice === null) {
40. $timeSlice = floor(time() / 30);
41. }
42. $secretkey = $this->debase32($secret);
43. $time = chr(0) . chr(0) . chr(0) . chr(0) . pack('N\*', $timeSlice);
44. $hm = hash\_hmac('SHA1', $time, $secretkey, true);
45. $offset = ord(substr($hm, -1)) & 0x0F;
46. $hashpart = substr($hm, $offset, 4);
47. $value = unpack('N', $hashpart);
48. $value = $value[1];
49. $value = $value & 0x7FFFFFFF;
50. $modulo = pow(10, $this->length);
51. **return** str\_pad($value % $modulo, $this->length, '0', STR\_PAD\_LEFT);
52. }
53. **public** **function** getQR($name, $secret, $title = null, $params = **array**()) {
54. $width = !empty**empty**($params['width']) && (int)$params['width'] > 0 ? (int)$params['width'] : 200;
55. $height = !empty**empty**($params['height']) && (int)$params['height'] > 0 ? (int)$params['height'] : 200;
56. $level = !empty**empty**($params['level']) && array\_search($params['level'], **array**('L', 'M', 'Q', 'H')) !== false ? $params['level'] : 'M';
57. $urlencoded = urlencode('otpauth://totp/' . $name . '?secret=' . $secret . '');
58. **if** (isset($title)) {
59. $urlencoded.= urlencode('&issuer=' . urlencode($title));
60. }
61. **return** 'https://chart.googleapis.com/chart?chs=' . $width . 'x' . $height . '&chld=' . $level . '|0&cht=qr&chl=' . $urlencoded . '';
62. }
63. **public** **function** verifyCode($secret, $code, $discrepancy = 1, $currentTimeSlice = null) {
64. **if** ($currentTimeSlice === null) {
65. $currentTimeSlice = floor(time() / 30);
66. }
67. **if** (strlen($code) != 6) {
68. **return** false;
69. }
70. **for** ($i = - $discrepancy;$i <= $discrepancy;++$i) {
71. $calculatedCode = $this->getCode($secret, $currentTimeSlice + $i);
72. **if** ($this->timingSafeEquals($calculatedCode, $code)) {
73. **return** true;
74. }
75. }
76. **return** false;
77. }
78. **public** **function** setCodeLength($length) {
79. $this->length = $length;
80. **return** $this;
81. }
82. **protected** **function** debase32($secret) {
83. **if** (empty**empty**($secret)) {
84. **return** '';
85. }
86. $base32chars =  **array**(
87. 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
88. 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
89. 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
90. 'Y', 'Z', '2', '3', '4', '5', '6', '7',
91. '=',
92. );
93. $base32charsFlipped = array\_flip($base32chars);
94. $paddingCharCount = substr\_count($secret, $base32chars[32]);
95. $allowedValues = **array**(6, 4, 3, 1, 0);
96. **if** (!in\_array($paddingCharCount, $allowedValues)) {
97. **return** false;
98. }
99. **for** ($i = 0;$i < 4;++$i) {
100. **if** ($paddingCharCount == $allowedValues[$i] && substr($secret, -($allowedValues[$i])) != str\_repeat($base32chars[32], $allowedValues[$i])) {
101. **return** false;
102. }
103. }
104. $secret = str\_replace('=', '', $secret);
105. $secret = str\_split($secret);
106. $binaryString = '';
107. **for** ($i = 0;$i < count($secret);$i = $i + 8) {
108. $x = '';
109. **if** (!in\_array($secret[$i], $base32chars)) {
110. **return** false;
111. }
112. **for** ($j = 0;$j < 8;++$j) {
113. $x.= str\_pad(base\_convert(@$base32charsFlipped[@$secret[$i + $j]], 10, 2), 5, '0', STR\_PAD\_LEFT);
114. }
115. $eightBits = str\_split($x, 8);
116. **for** ($z = 0;$z < count($eightBits);++$z) {
117. $binaryString.= (($y = chr(base\_convert($eightBits[$z], 2, 10))) || ord($y) == 48) ? $y : '';
118. }
119. }
120. **return** $binaryString;
121. }
122. **private** **function** timingSafeEquals($safeString, $userString) {
123. **if** (function\_exists('hash\_equals')) {
124. **return** hash\_equals($safeString, $userString);
125. }
126. $safeLen = strlen($safeString);
127. $userLen = strlen($userString);
128. **if** ($userLen != $safeLen) {
129. **return** false;
130. }
131. $result = 0;
132. **for** ($i = 0;$i < $userLen;++$i) {
133. $result|= (ord($safeString[$i]) ^ ord($userString[$i]));
134. }
135. **return** $result === 0;
136. }
137. }
138. ?>

Saat user selesai melakukan login dan berhasil login maka sistem tidak langsung mengarahkan ke dashboard, tetapi akan diarahkan untuk memasukan kode OTP. jika saat diarahkan untuk input kode OTP data user tidak memiliki secret key TOTP maka sistem akan membangkitkan secret key dan menirimkan qr-code utnuk dapat di scan dan dibaca oleh andoid untuk menghasilkan kode OTP. jika kode sesuai maka akan diarahkan ke menu dashboard. Proses pembuatan secret key kode TOTP hanya sekali saat registrasi awal. Untuk melakukan reset dapat melalui operator atau menu reset dengan menggunakan email konformasi.

## Hasil pembuatan model TOTP dan Algoritma rsa

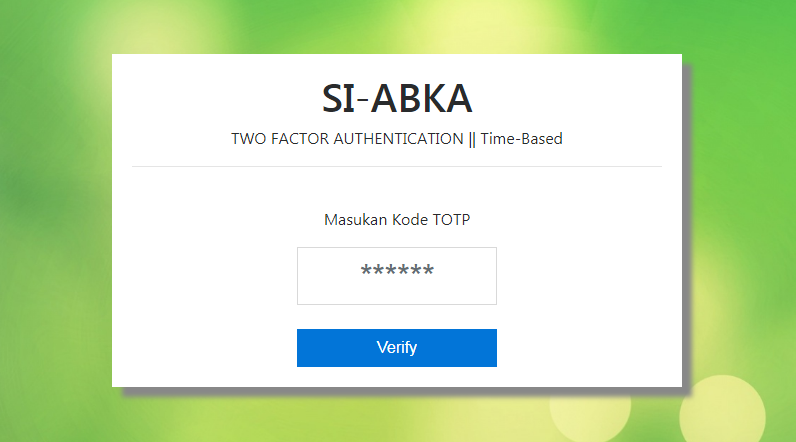
Pada sub bab ini menjelaskan alur mulai dari pembangkitan pasangan kode secret dan public algoritma RSA, pembangkitan secret key TOTP, dan penangan saat terjadi nya error.

* 1. Saat user akan masuk kesistem tampilan website seperti pada umumnya yaitu memasukan username dan password. pada SI-Abka username digantikan dengan KTA dan password seperti pada sistem lain. Seperti pada tampilan dibawah ini.



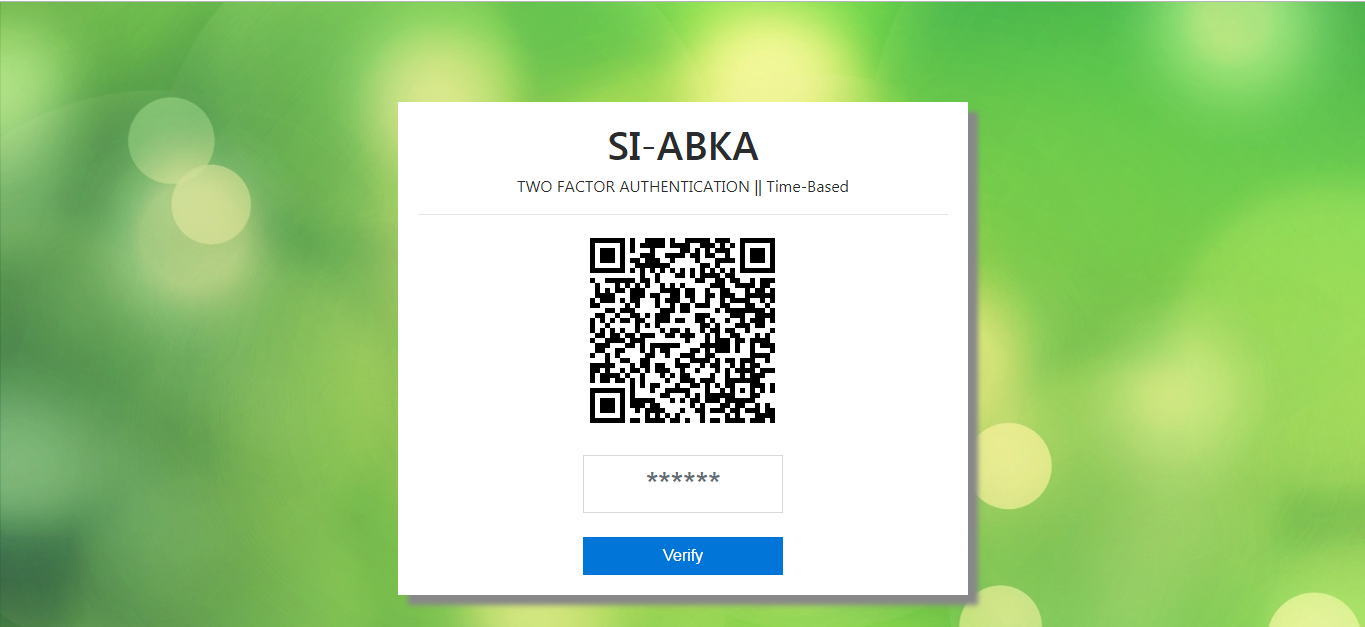
Gambar 4.3. tampilan login SI-Abka

* 1. Setelah memasukan KTA maka data user akan di cek di database. Jika user memiliki seret key akan diarahkan ke tampilan seperti pada gambar dibawah ini. Maka user tinggal membuka aplikasi TOTP dan memasukan angka yang di tampilkan di aplikasi tersebut.



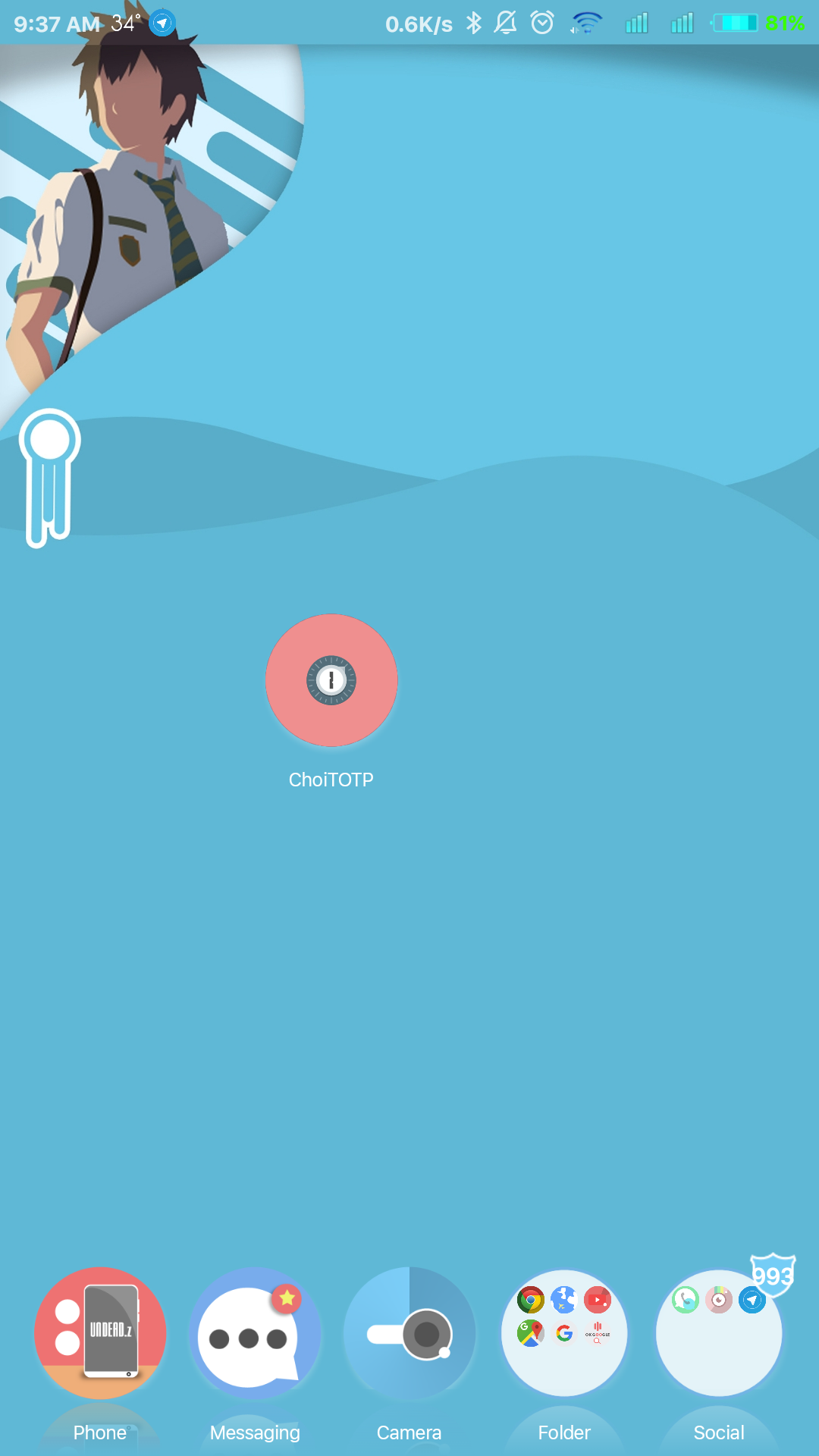
Gambar 4.4. input kode TOTP

* 1. Jika user belum meimiliki secret key maka sistem akan mengarahkan usr ke tampilan generate secret key seperti di bawah ini



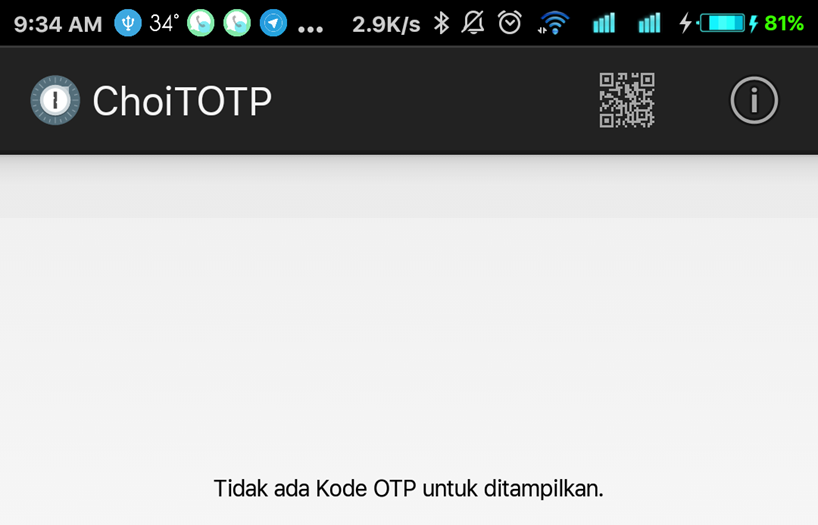
Gambar 4.5. input kode TOTP dengan qr-code

* 1. Untuk mendapatkan kode OTP maka user perlu melakukan scan qr-code terlebih dahulu seperti tampilan di bawah ini



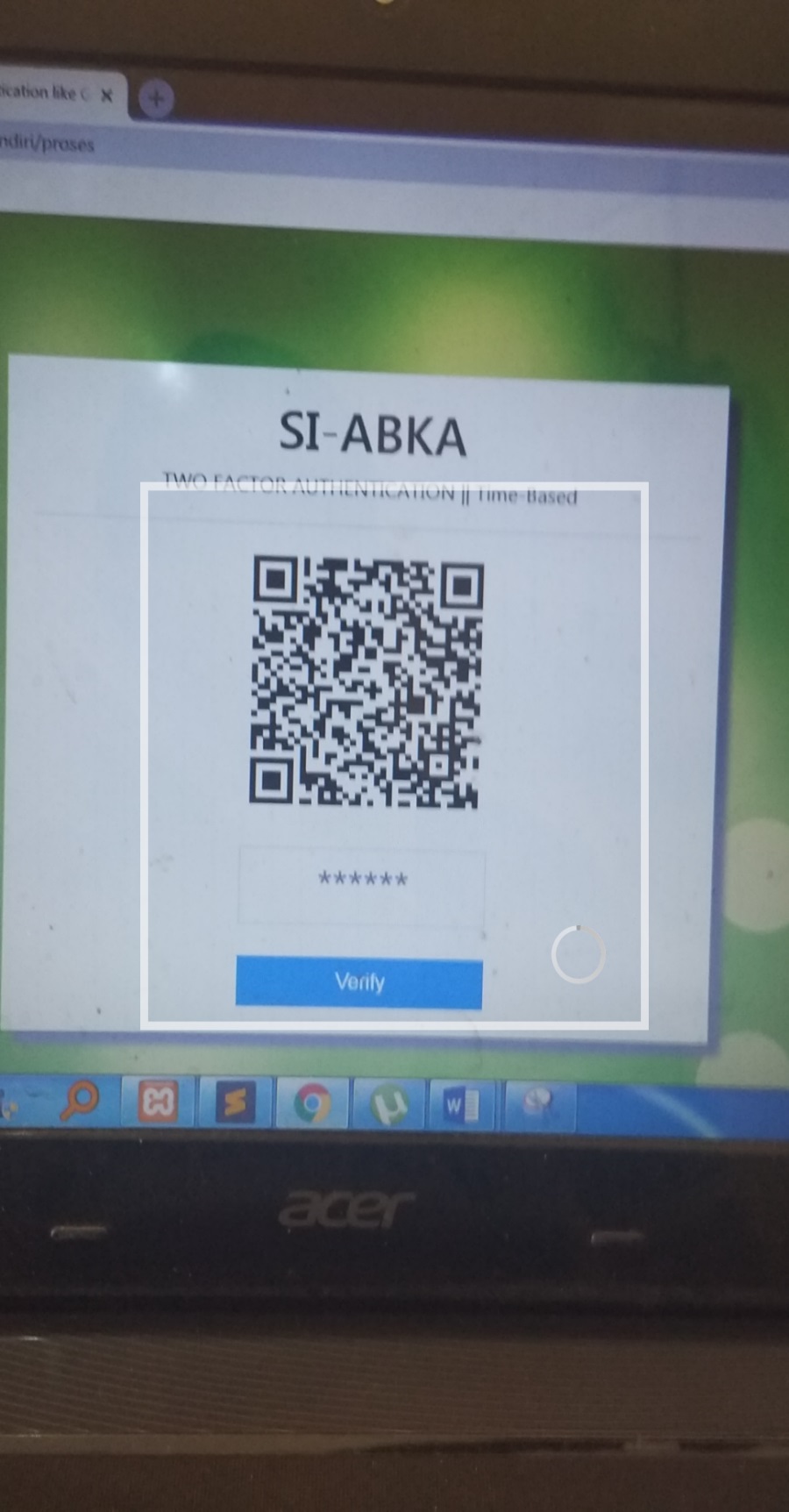
Gambar 4.6. aplikasi ChoiTOTP

* 1. Setelah aplikasi terbuka user perlu memasukan secret key dengan memindai qr-code. Klik pada icon di pojok kanan atas.



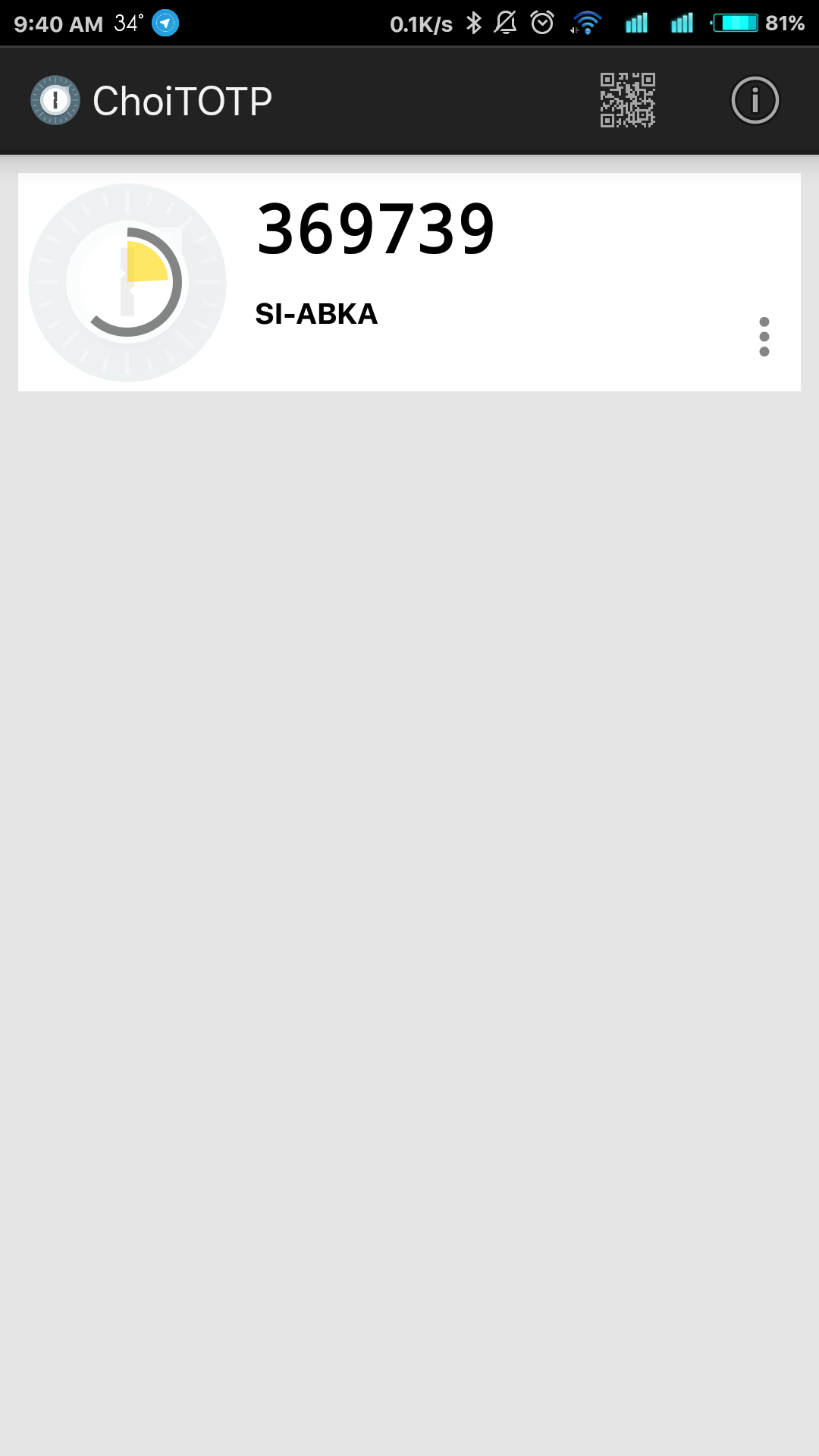
Gambar 4.7. menu aplikasi ChoiTOTP

* 1. Aplikasi akan mengaktifkan kamera belakang dan akan user perlu mengarahkan kamera ke qrcode seperti pada tampilan seperti gambar 4.5



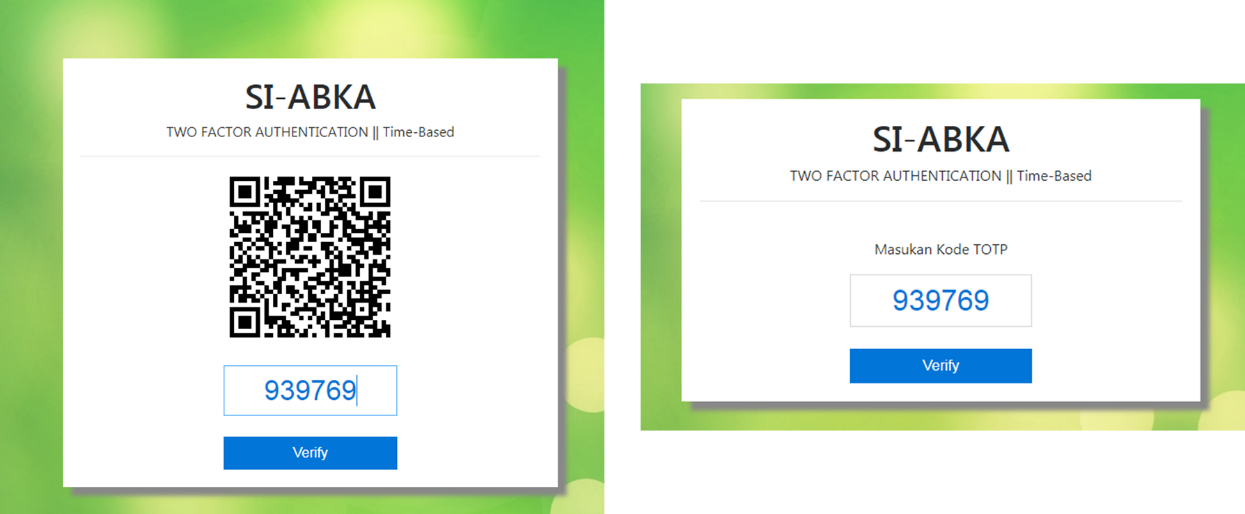
Gambar 4.8. tampilan scan qr-code

* 1. Setelah berhasil di pindai maka aplikasi akan menampilkan enam digit angka yang dapat digunakan untuk login seterusnya.



Gambar 4.9.kode TOTP berhasil didapatkan

* 1. Enam digit angka tersebut daat digunaka pada form input seperti pada gambar 4.4 dan gambar 4.5. Angka tersebut akan berubah setiap 30 detik.

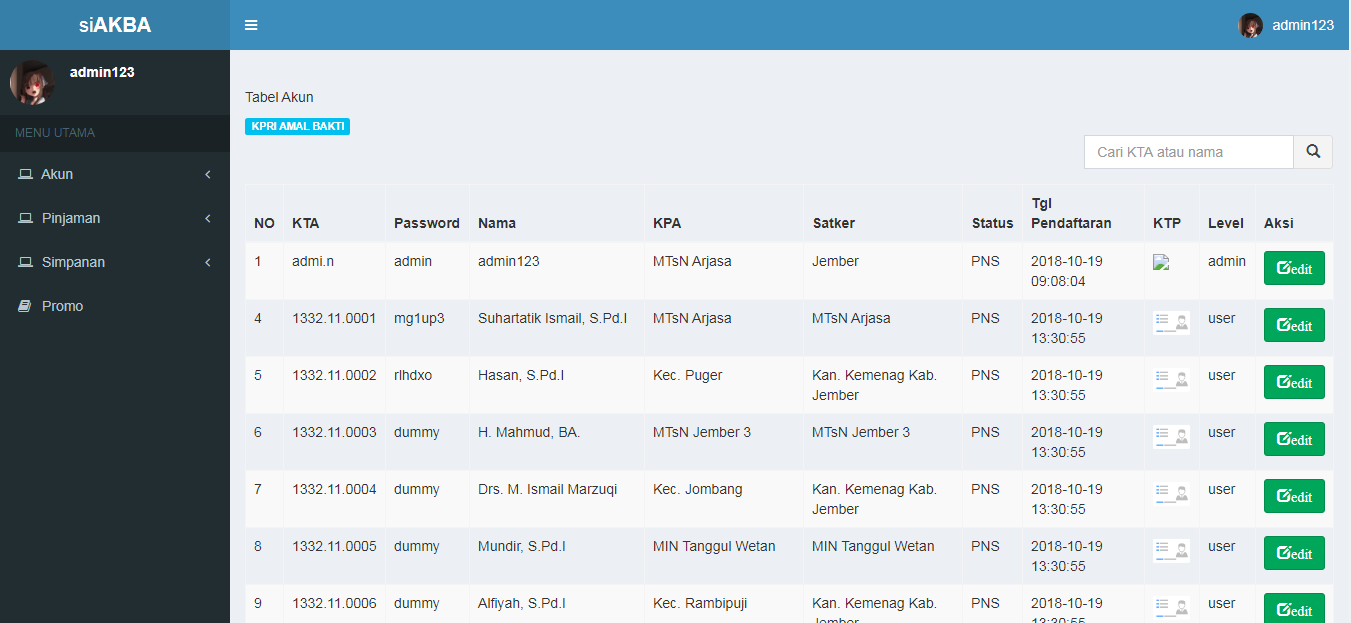


Gambar 4.10. memasukan kode TOTP

* 1. Jika kode tersebut cocok maka sistem akan mengarahkan ke dashboard masing-masing sesuai dengan levelnya

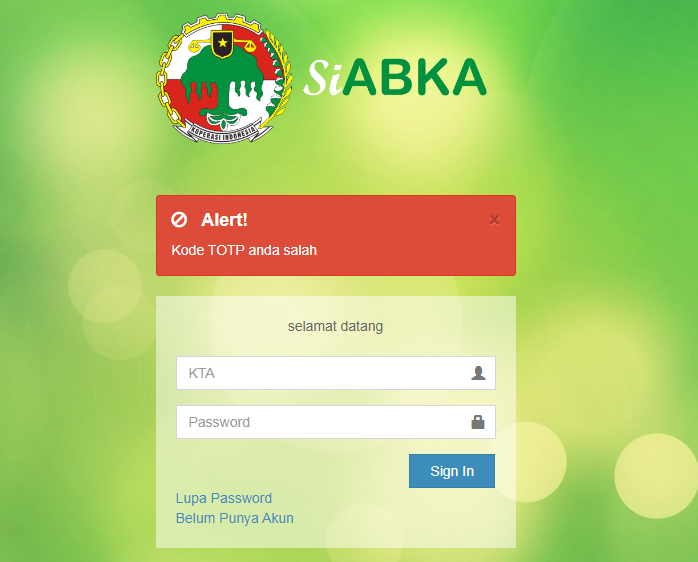


Gambar 4.11.dasboard user



Gambar 4.12. dashboard admin

* 1. Jika kode tersebut tidak cocok maka user akan di arahkan kembali ke tampilan login dan sistem menampilkan pemberitahuan bahwa kode tersebut salah.



Gambar 4.13. pemberitahuan kode OTP salah

## Hasil pengujian uji keamanan

Pengujian keamanan ini berfungsi untuk mengetahui tingkat keamanan jika web sistem SI-abka diserang. Pengujian ini menggunakan dua kasus yaitu antara SI-abka yang mengimplementasikan *two factor authentication* dan algortima RSA, dengan SI-abka tanpa tambahan modul tersebut.

### *Pengujian* *brute force*

Pengujian *brute force* di lakukan dengan cara mencoba setiap kemungkinan kode OTP. Kode OTP yang di bangkitkan memiliki kriteria 6 digit dan merupakan bilangan cacah atau bilangan postif dan nol. Total kemungkinan variasi kode OTP berjumlah satu juta mulai dari 000000 – 999999.

Aplikasi burpsuite bekerja dengan cara mengirimkan file header yang berisi data akun berupa KTA, password, dan kode OTP. Aplikasi burpsuite akan membantu pengujian dengan mengacak kode OTP dan mengirimkanya ke sistem. Pengujian kode OTP sukses jika mendapatkan respons code 303 dan gagal jika mendapatkan respons code 401. Respons code 303 berarti mengalihkan aplikasi web ke url tertentu (beranda user) sedangkan 401 berarti halaman yang sedang di akses tidak dapat dimuat sampai user login dengan akun yang valid.

Pengujian kali ini menggunakan localhost dengan spesikasi alat sebagai berikut:

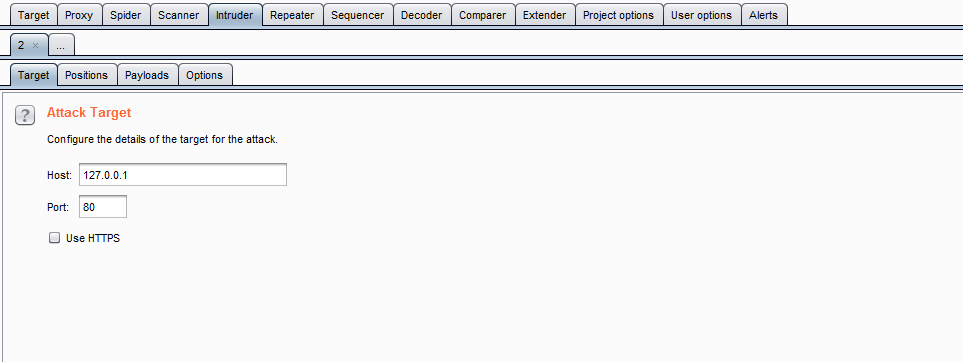
1. Prosesor : Intel Core i3 2330M processor (3MB L3 cache, 2.20 GHz)
2. Sistem operasi : Windows 7 Ultimate
3. Memori : 4 GB DDR3 SDRAM 1066 MHz
4. Grafis : Intel HD 3000 Graphics

Sebelum melakukan pengujian *brute force* terlebih dahulu menyiapkan data-data pendukung seperti *host target*, dan *header post*. Untuk mendapatkan data tersebut dapat menggunakan burpsuite dengan beberapa pengaturan terlebih dahulu.

Contoh file header post pengiriman data login

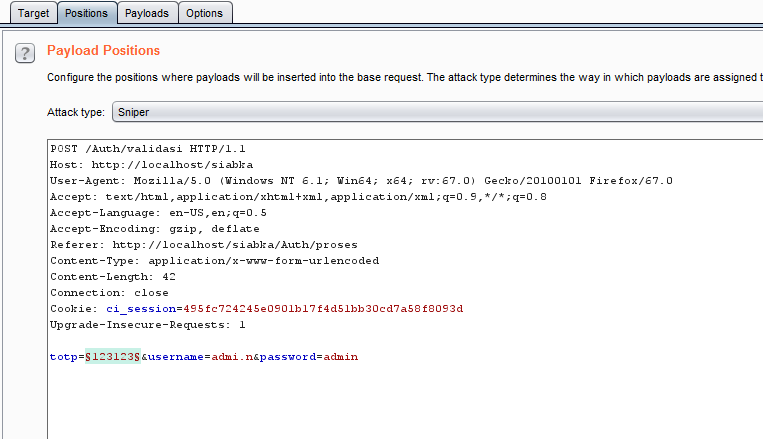
1. POST /Auth/validasi HTTP/1.1
2. Host: localhost
3. UserAgent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:67.0) Gecko/20100101 Firefox/67.0
4. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
5. Accept-Language: en-US,en;q=0.5
6. Accept-Encoding: gzip, deflate
7. Referer: http://localhost/siabka/Auth/proses
8. Content-Type: application/x-www-form-urlencoded
9. Content-Length: 42
10. Connection: close
11. Cookie: ci\_session=495fc724245e0901b17f4d51bb30cd7a58f8093d
12. Upgrade-Insecure-Requests: 1
14. totp=123123&username=admi.n&password=admin

setelah mendapatkan file header login pilih tab intruder dan mulai isi variabel-variabel target



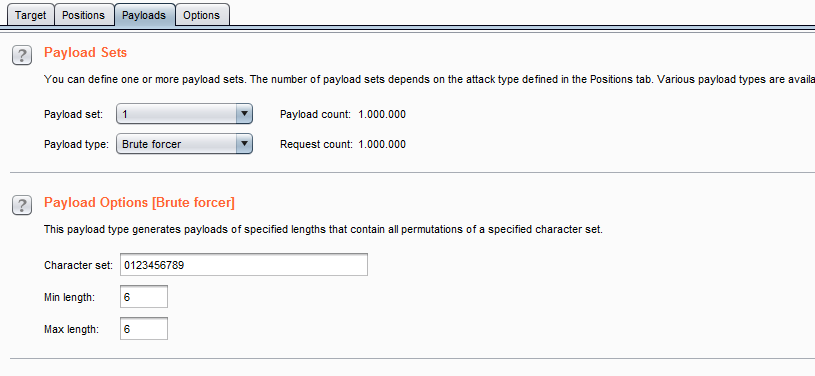
Gambar 4.14. penentuan target sistem

Pada tab *positions* diisi dengan file *header* yang telah di dapatkan tadi



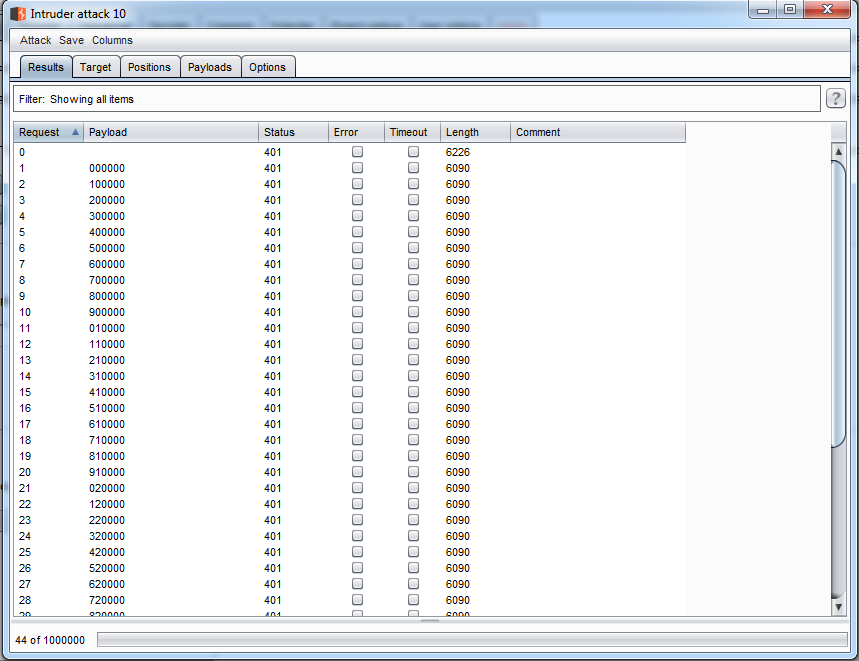
Gambar 4.15. variabel header

Pada tab *payloads* berisi aturan *brute force* yang akan diisi ke dalam header. Pada pengujian kali ini memerlukan 6 digit angka bilangan cacah



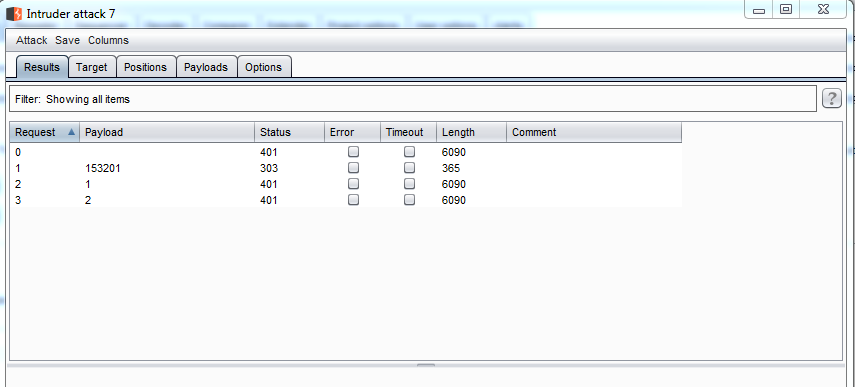
Gambar 4.16. variabel payload brute force

Jika seluruh data sudah sesuai maka klik start attack. Berikut hasil *brute force*  dalam 30 detik pertama.



Gambar 4.17. hasil brute force

Dalam 30 detik burpsuite berhasil mengirimkan 44 file header yang telah di modifikasi. Dari 44 header yang dikirim semua di tolak oleh sistem ditandai dengan status 401. Jika kode OTP benar makan akan mendaatkan respons code 303 seperti di bawah ini

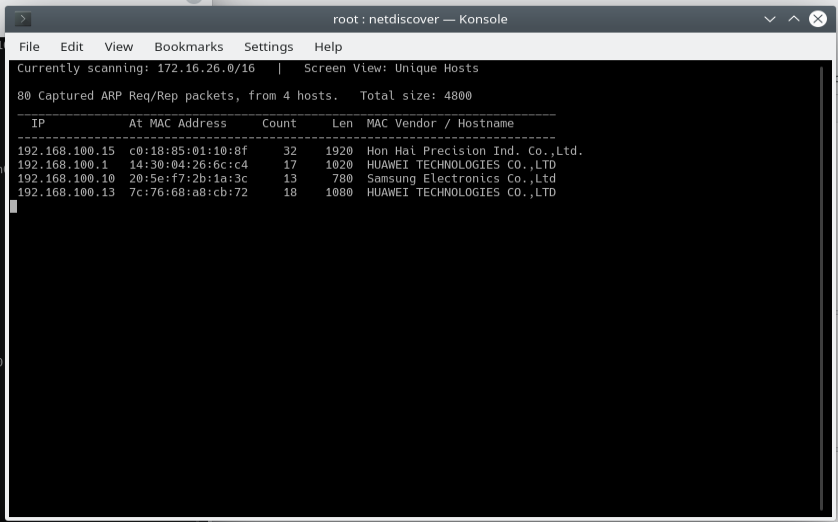


Gambar 4.18. respons code OTP benar

Pengujian dilakukan hanya dalam 30 detik karena setiap 30 detik kode OTP tersebut akan berganti secara terus menerus. Dan untuk pengujian kali ini menggunakan localhost karena saat menggunakan hosting di internet maka akan lebih lambat lagi dalam pengiriman header tersebut dikarenakan spesifikasi server tidak sama. Dari pengujian tersebut didapat dalam 30 detik burpsuite hanya mampu mengirimkan 44 header dari 1 juta kemungkinan header yang ada. Hal ini berarti hanya 0.044 % kemungkinan kode OTP tersebut benar dalam 30 detik. Presentase kemungkinan sangat kecil untuk dapat ditebak dan dapat dikatakan sangat aman. Banyaknya header yang dikirim juga dapat dipengaruhi oleh spesifikasi dari komputer penyerangan dan server target.

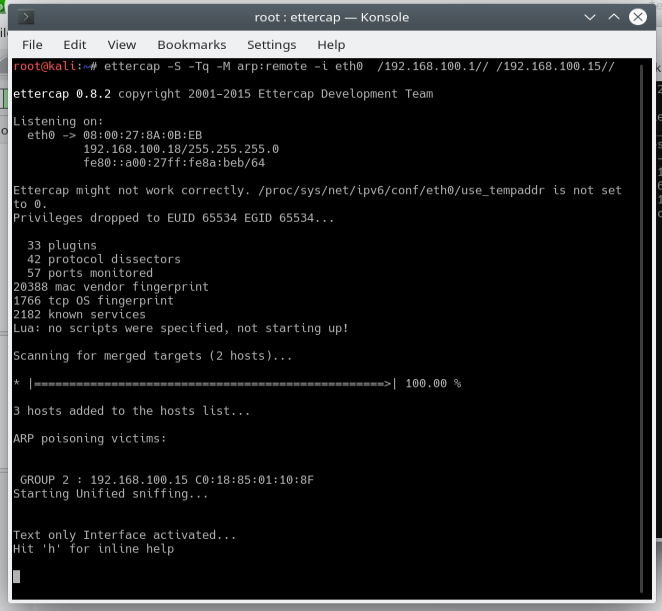
### Pengujian MITM

Pengujian teknik MITM dapat dilakukan dengan syarat target dan penyerang berada dalam satu jaringan lokal. Setelah yakin kedua device terkoneksi dalam satu jaringan yang sama maka penyerang dapat mencari ip target dengan bantuan tools netdiscover .



Gambar 4.19. proses scan ip target

Setelah mendapatkan ip target maka kita perlu mengelabui router dan target dengan menggunakan tools ettercap



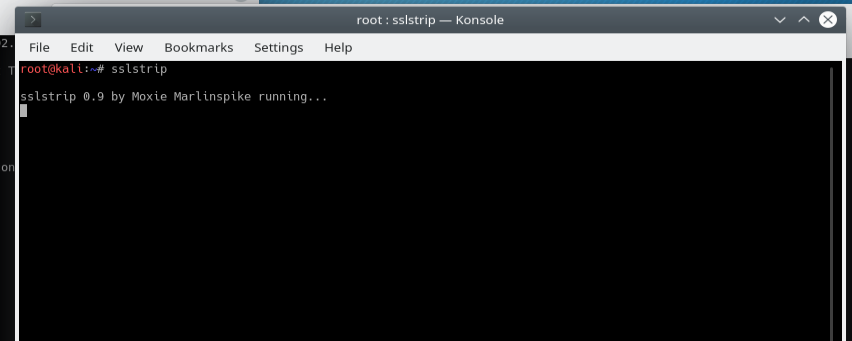
Gambar 4.20. arp poisoning ip

Dalam tahap ini kita sudah berhasil berada di tengah-tengah target dan router. Agar router dan target dapat berkomunikasi maka kita perlu menyalurkan datanya sekaligus dapat membaca arus data. Untuk dapat menyalurkan data kita perlu melakukan routing terlebih dahulu



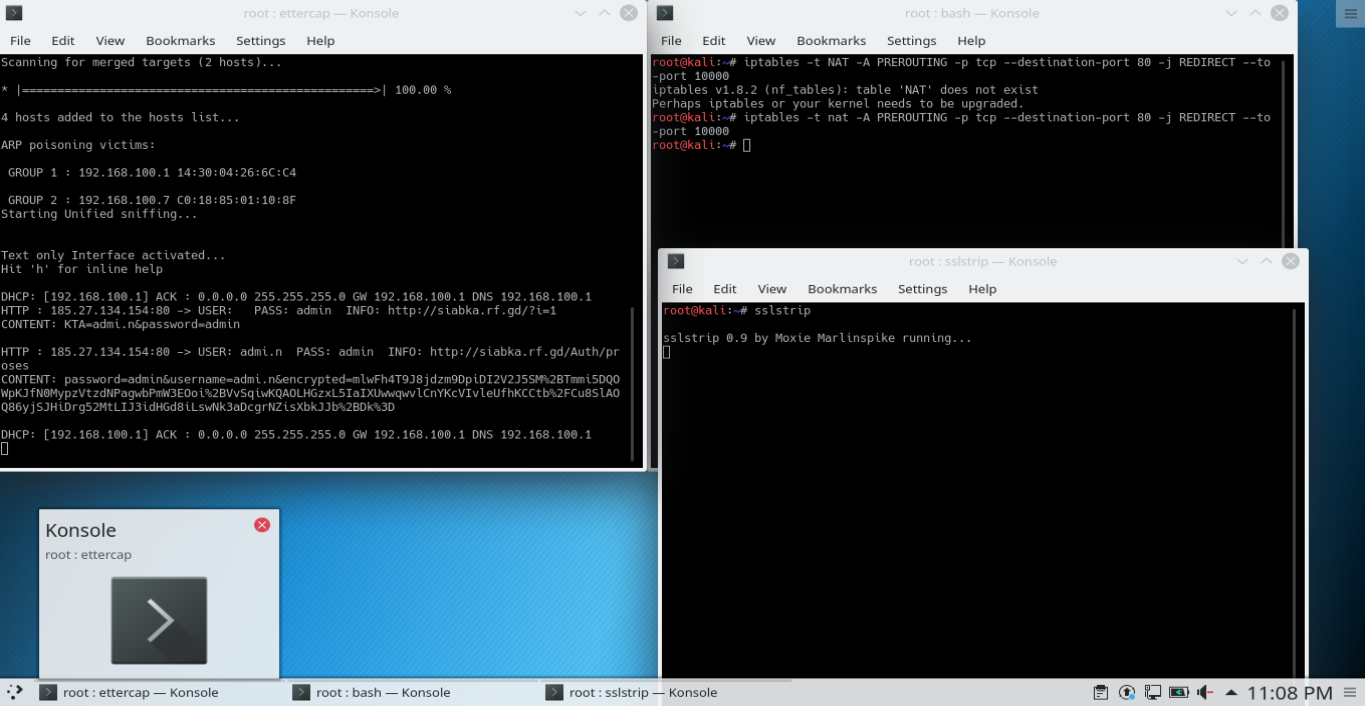
Gambar 4.21. pembelokan data komunikasi dengan router

setelah target dan router dapat saling berkomunikasi kita perlu membelokan traffict https ke http agar dapat mudah dibaca oleh wireshark. Maka kita membutuhkan tools sslstrip.



Gambar 4.22. dekripsi ssl

Dalam tahap ini kita tinggal menunggu target melakukan pengiriman data ke server. Contoh hasil pembacaan data dengan teknik MITM.



Gambar 4.23. hasil pembacaan packet data

Dari hasil pembacaan diatas dapat kita lihat bahwa username , password, dan kode otp dapat dilihat atau dibaca. Tetapi kode OTP telah di enkripsi dengan algoritma RSA sehingga tidak mudah dibaca oleh penyerang.

# PENUTUP



## Kesimpulan

Hasil dari penelitian yang dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Untuk meningkatkan otentikasi sistem SI-Abka diperlukan faktor tambahan agar tidak sembarangan orang dapat masuk ke sistem SI-Abka. Faktor tambahan tersebut adalah menggunakan metode OTP. metode OTP yang digunakan memakai metode TOTP (*Time One Time Password*). Penggunaan TOTP dipilih karena tidak memakan resource yang banyak dan tidak membebani sistem saat menangani arus data yang besar. Resource adalah hal ini meliputi spesifikasi server dan penggunaan pihak ketiga seperti sms gateway dan biaya server tambahan.
2. Penggunaan Two Factor Authentication yang menggunakan metode TOTP memiliki output berupa secret key dan kode OTP yang dibangkitkan. Kedua data tersebut sangat penting dalam pengamanan data SI-Abka, oleh karena itu diperlukan enkripsi khusus untuk mengamankan data yang dikirim oleh server ke client dan sebaliknya. penggunaan algoritma RSA diperlukan untuk mengamankan kedua data tersebut. Dengan adanya algoritma RSA maka data tersebut akan lebih aman karena data tidak bisa dibaca dan perlu melakukan dekripsi untuk membacanya.
3. Pembangkitan kode OTP menggunakan metode *time one time password* (TOTP). Metode tersebut menggunakan factor bergerak yaitu faktor waktu dan secret key khusus. Faktor waktu di perlukan untuk membangkitkan kode OTP di sisi server maupun client agar sama atau identik, sedangkan secret key digunakan sebagai pembeda tiap user agar kode OTP yang dibangitkan tidak sama. proses pengamanan autentikasi SI-Abka di mulai saat user akan masuk ke dalam sistem. Agar user dapat masuk ke dalam sistem di perlukan kode OTP. kode OTP didapatkan melalui aplikasi khusus yang dapat di download di playstore atau aplikasi yang dibuat oleh peneliti. Setelah user memasukan kode OTP ke dalam form login makan sever akan merespon apakah kode tersebut sama dengan kode yang dibangkitkan oleh server. Faktor waktu sangat berpengaruh dalah pembangkitan kode OTP, oleh karena itu waktu di server dan waktu di client harus sama.
4. Data yang dikirim oleh server maupun client memiliki risiko untuk di lihat atau disadap oleh orang yang tidak bertanggup jawab. Dari masalah tersebut diperlukan pengamanan tambahan yaitu enkripsi menggunakan merode RSA. Proses pertama adalah membuat secret key untuk setiap akun yang ada, secret key diperlukan untuk mengamankan kode OTP untuk menghidari kode OTP yang diinputkan user dicuri. Secret key dibuat secara acak dan agar secret key disimpan user perlu melakukan pemindaian qrcode yang dihasilkan sistem. Setelah qrcode dipindai maka akan muncul kode khusus yang dapat di inputkan ke sistem. Jika kode tersebut sesuai maka sistem akan menampilkan beranda sesuai kategori user tersebut. Kode OTP yang diinputkan perlu untuk di enkripsi karena kode tersebut akan valid selama 30 detik setelah dibangkitkan. Dengan mengkripsi kode OTP maka penyerang tidak dapat membaca kode tersebut dan keaman data dapat terjamin
5. Dalam penilitian kali ini peneliti melakukan pengukuran tingkat keamanan dari penggunaan two factor autentication . percobaan oengukuran tingkat keaamanan menggunaka dua teknik yaitu brute force dan man in the middle (MITM). Percobaan tes keamanan sistem dengan teknik brute force menghasilkan dari satu juta kombinasi komputer peneliti hanya dapat mampu mengirimkan 44 kombinasi dalam 30 detik pertama dan 30 detik berikutnya kode OTP tersebut akan berganti. Presentase kemungkinan tertebaknya kode OTP dalam percobaan tersebut hanya sebesar 0.044 % . dalam percobaan dengan teknik man in the middle peneliti berhasil mendapatkan data yang dikirim oleh client ke server. Data yang didapat berupa nomor KTA, password, dan kode OTP yang valid. Dari ketiga data tersebut kode OTP berhasil di enkripsi dan tidak dapat dibaca oleh peneliti. Kode OTP hanya bisa di baca setelah didekripsi oleh server. Dari percobaan tersebut keamanan data yang dikirim oleh client ke server bisa dikatakan aman

## Saran

Pada sub bab ini akan dijabarkan beberapa saran penilitian untuk penelitian lain yang serupa atau penambahan fitur yang berhubungan dengan aplikasi implementasi *Two Factor Authentication* pada SI-Abka adalah sebagai berikut :

1. Modul yang sudah dibuat masih dapat dimodifikasi agar dapat mengikuti perkembangan proses bisnis perusahaan, modifikasi tersebut bertujuan agar modul tersebut dapat digunakan / diimplementasikan pada sistem yang sudah ada.
2. dibutuhkannya fitur backup agar data-data yang sudah tersimpan lebih aman jika terjadi hal-hal yang tidak diinginkan seperti sistem down, server rusak dll. sehingga file backup dapat digunakan untuk melanjutkan proses bisnis kembali tanpa kehilangan waktu untuk menulis kembali data - data yang hilang.
3. Dalam segi antar muka, modul login masih sederhana dan harus mendapatkan desain yang lebih baik, seperti penempatan tombol dan teks pembantu. Tampilan utama bagi user harus menarik dari segi desain, warna, dan lain - lain.

# Daftar pustaka

Inayatullah. (2007). Analisis Penerapan Algoritma MD5 Untuk Pengamanan Password. *jurnal ilmiah STMIK GI MDP*, *3*(3), 1–5.

Khairina, D. M. (2011). ANALISIS KEAMANAN SISTEM LOGIN. *Jurnal Informatika Mulawarman*, *6*, 64–67.

Kim, H., Lee, Y.-G., Lee, K.-S., & Jun, M.-S. (2009). Design and Implementation of Multi Authentication Mechanism for Secure Electronic Commerce. *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, 215–219. https://doi.org/10.1109/SNPD.2009.70

M’Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). *TOTP: Time-Based One time password Algorithm* (No. RFC6238). https://doi.org/10.17487/rfc6238

Muchlis, B. S., Budiman, M. A., & Rachmawati, D. (2007). Teknik Pemecahan Kunci Algoritma Rivest Shamir Adleman(RSA) dengan Metode Kraitchik. *jurnal & Penelitian Teknik Informatika*, *2*, 2.

Musliyana, Z., Arif, T. Y., & Munadi, R. (2016). Peningkatan Sistem Keamanan Autentikasi Single Sign On (SSO) Menggunakan Algoritma AES dan *One time password* Studi Kasus: SSO Universitas Ubudiyah Indonesia. *Jurnal Rekayasa Elektrika*, *12*(1), 21. https://doi.org/10.17529/jre.v12i1.2896

Mustofa, R. P. (2003). *APLIKASI MOBILE ANDROID “ONE TIME PASSWORD(OTP)” UNTUK MENINGKATKAN KEAMANAN OTENTIKASI* (hlm. 1–15). yogyakarta: SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER A MIKOM YOGYAKARTA.

Pramudita, K. E. (2010). *Brute force* Attack dan Penerapannya pada Password Cracking. *Makalah IF3051 Strategi Algoritma*, *sem 1*.

Rosnawan, D. (2011). Aplikasi Algoritma RSA untuk Keamanan Data pada Sistem Informasi Berbasis Web. *Universitas Negeri Semarang.*, 1–25.

Sardju, E. R., Magdalena, Ir. R., & Atmaja, R. (2015). IMPLEMENTASI ALGORITMA RSA UNTUKENKRIPSI DAN DEKRIPSI SMS (SHORT MESSAGE SERVICE) PADA PONSEL BERBASIS ANDROID. *e-Proceeding of Engineering*, *2*, 2435.

Ungkawa, U., Dewi, I. A., & Putra, K. R. (t.t.). IMPLEMENTASI ALGORITMA TIME-BASED *ONE TIME PASSWORD* DALAM OTENTIKASI TOKEN INTERNET BANKING. *nstitut Teknologi Nasional Bandung*, 1–10.

# LAMPIRAN