BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dibahas metodologi yang digunakan untuk mengimplementasikan *Elliptic Curve Digital Signature Algorithm* (ECDSA) dan *Secure Hash Algorithm 256* (SHA 256) pada login akun dengan sistem *Two-Factor Authentication* (2FA) yang meliputi identifikasi masalah, model dasar, pengembangan model, konstruksi program, *library* Python yang digunakan, proses validasi program, dan pengambilan kesimpulan.

3.1 Identifikasi Masalah

Keamanan autentikasi pengguna adalah aspek penting dalam menjaga integritas dan kerahasiaan sistem informasi. Sistem autentikasi konvensional yang hanya mengandalkan *username* dan *password* rentan terhadap berbagai serangan seperti *phishing*, *brute-force*, dan pencurian kata sandi. Untuk meningkatkan keamanan, sistem 2FA digunakan dengan mengombinasikan sesuatu yang pengguna ketahui (*password*) dan sesuatu yang pengguna miliki (kunci privat).

Dalam hal ini, penerapan ECDSA dan SHA 256 menawarkan solusi lebih aman melalui tanda tangan digital unik yang sulit dipalsukan. Dengan ECDSA, pengguna menggunakan kunci privat untuk menandatangani pesan sebagai faktor kedua autentikasi, sehingga hanya pemilik sah kunci privat yang dapat mengakses sistem.

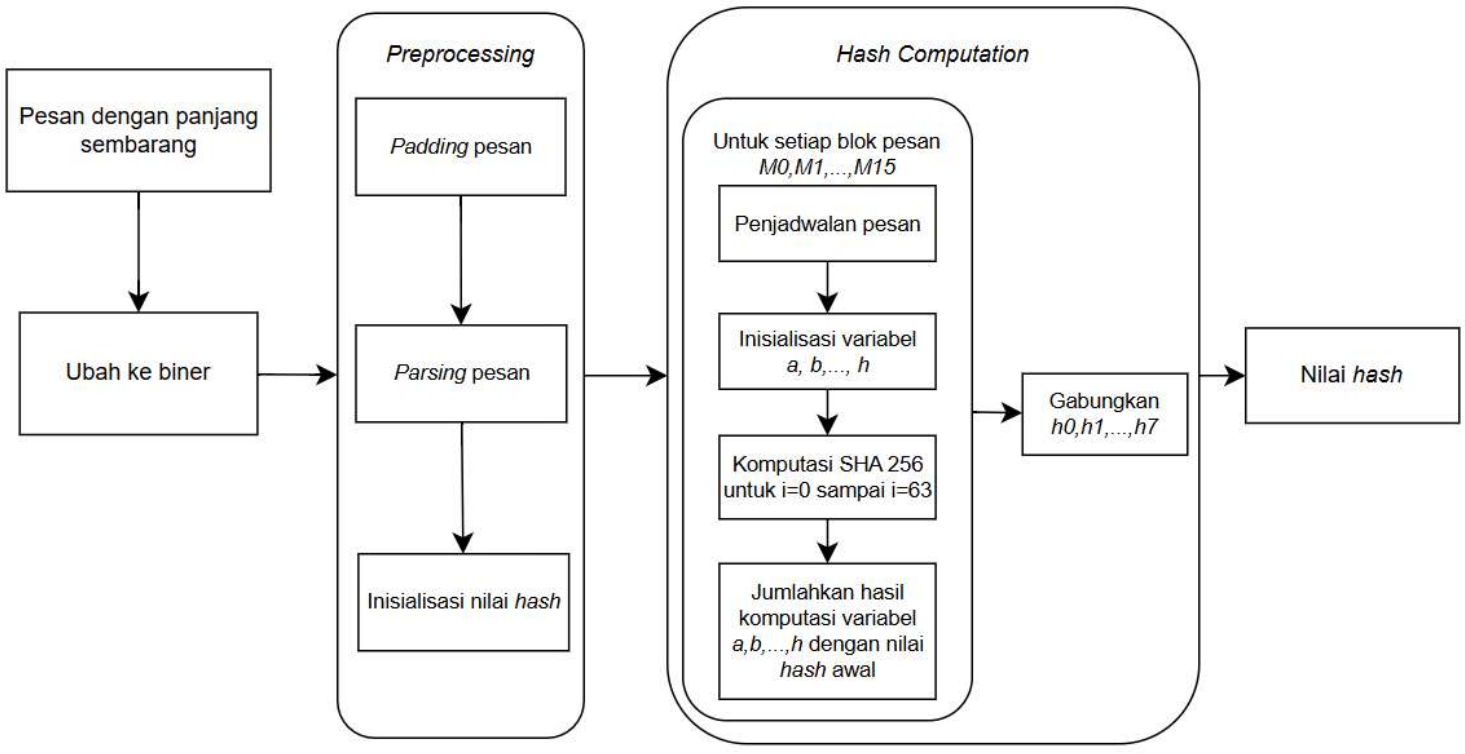
Dalam mengembangkan sistem 2FA berbasis ECDSA dan SHA 256 ini, salah satu tantangan yang muncul adalah menjaga keamanan kunci privat yang menjadi komponen krusial dalam autentikasi. Diperlukan mekanisme penyimpanan yang dapat melindungi kunci privat dari akses tidak sah baik di sisi *client* maupun *server*, karena penyimpanan kunci yang tidak aman berpotensi membuka celah bagi pencurian data autentikasi.

Selain terkait kunci privat, dalam ECDSA penting sekali untuk dapat memverifikasi tanda tangan digital secara tepat. Sistem perlu memastikan bahwa tanda tangan digital yang dihasilkan oleh kunci privat pengguna dapat diverifikasi secara akurat oleh *server*. Ketepatan ini penting agar hanya pengguna yang sah yang dapat lolos autentikasi, tanpa risiko kesalahan atau manipulasi tanda tangan.

3.2 Model Dasar

3.2.1 Skema SHA 256

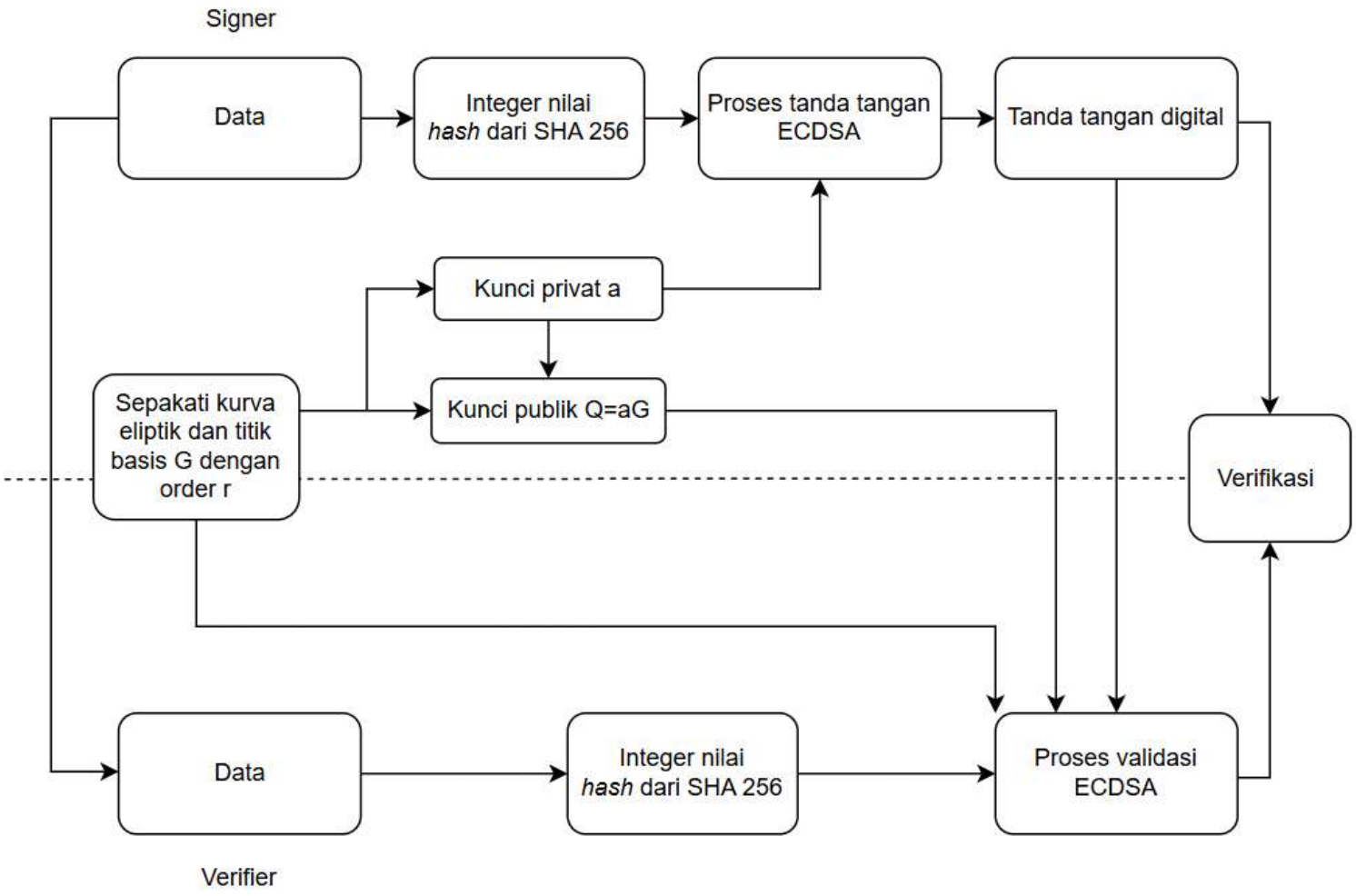
SHA 256 digunakan untuk menghasilkan nilai hash unik dari sebuah input data. Pada penelitian kali ini, SHA 256 digunakan untuk menyimpan input dari *client* berupa *username*, *password*, dan kunci privat secara aman. Skema SHA 256 terdiri dari tiga proses yaitu mengubah pesan ke dalam bentuk biner, melakukan *preprocessing*, dan melakukan *hash computation*. Skema SHA 256 dapat dilihat pada Gambar 3.1.



Gambar 3.1 Skema SHA 256

3.2.2 Skema ECDSA

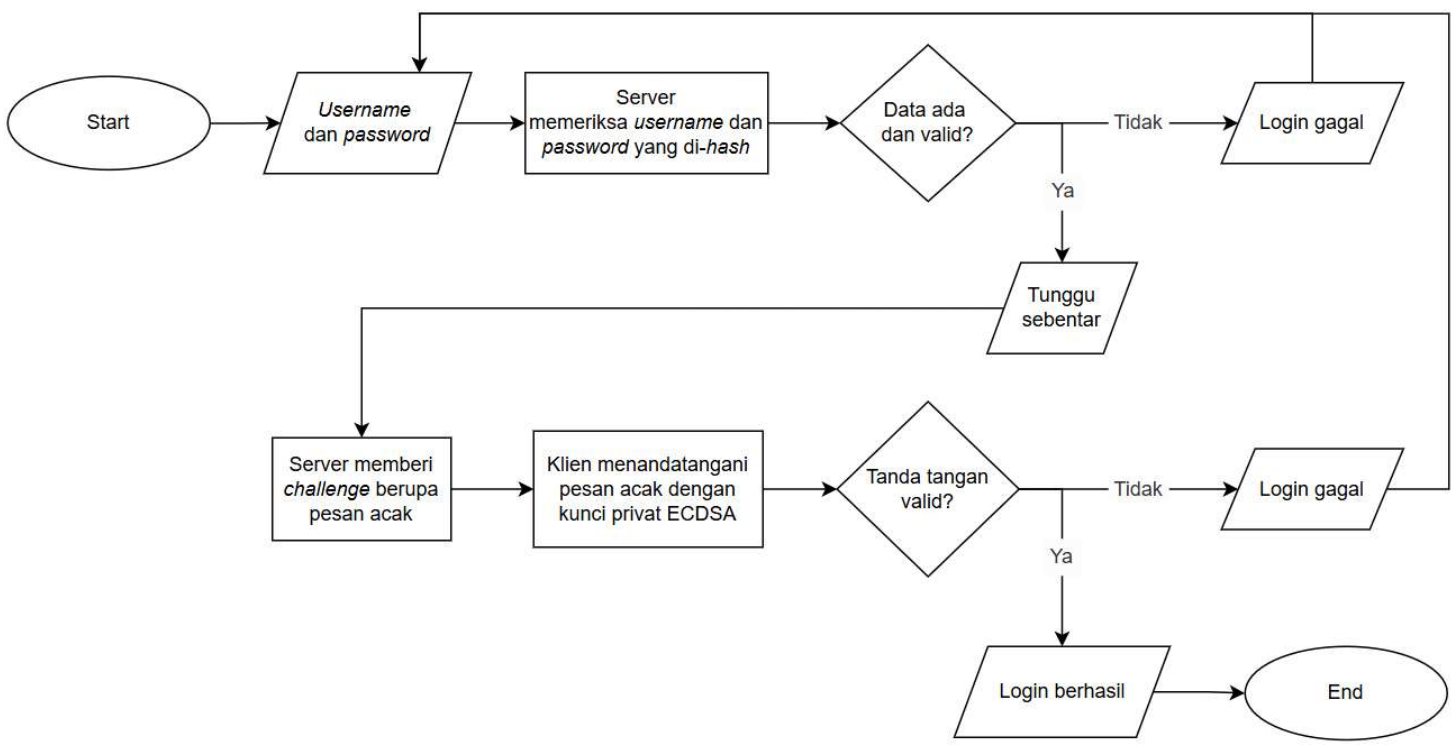
ECDSA adalah algoritma tanda tangan digital berbasis kriptografi eliptik yang digunakan untuk memastikan autentikasi dan integritas data. Pada penelitian ini, ECDSA digunakan untuk memverifikasi identitas pengguna dalam skema 2FA. ECDSA digunakan agar data yang diunggah ke *server* merupakan data yang valid dan telah diuji kebenarannya melalui berbagai rangkaian proses. ECDSA menggunakan fungsi *hash* untuk mengamankan data pengguna, di mana pada penelitian ini menggunakan SHA 256. Skema ECDSA dapat dilihat pada Gambar 3.2.



Gambar 3.2 Skema ECDSA

3.2.3 Bagan Alur 2FA

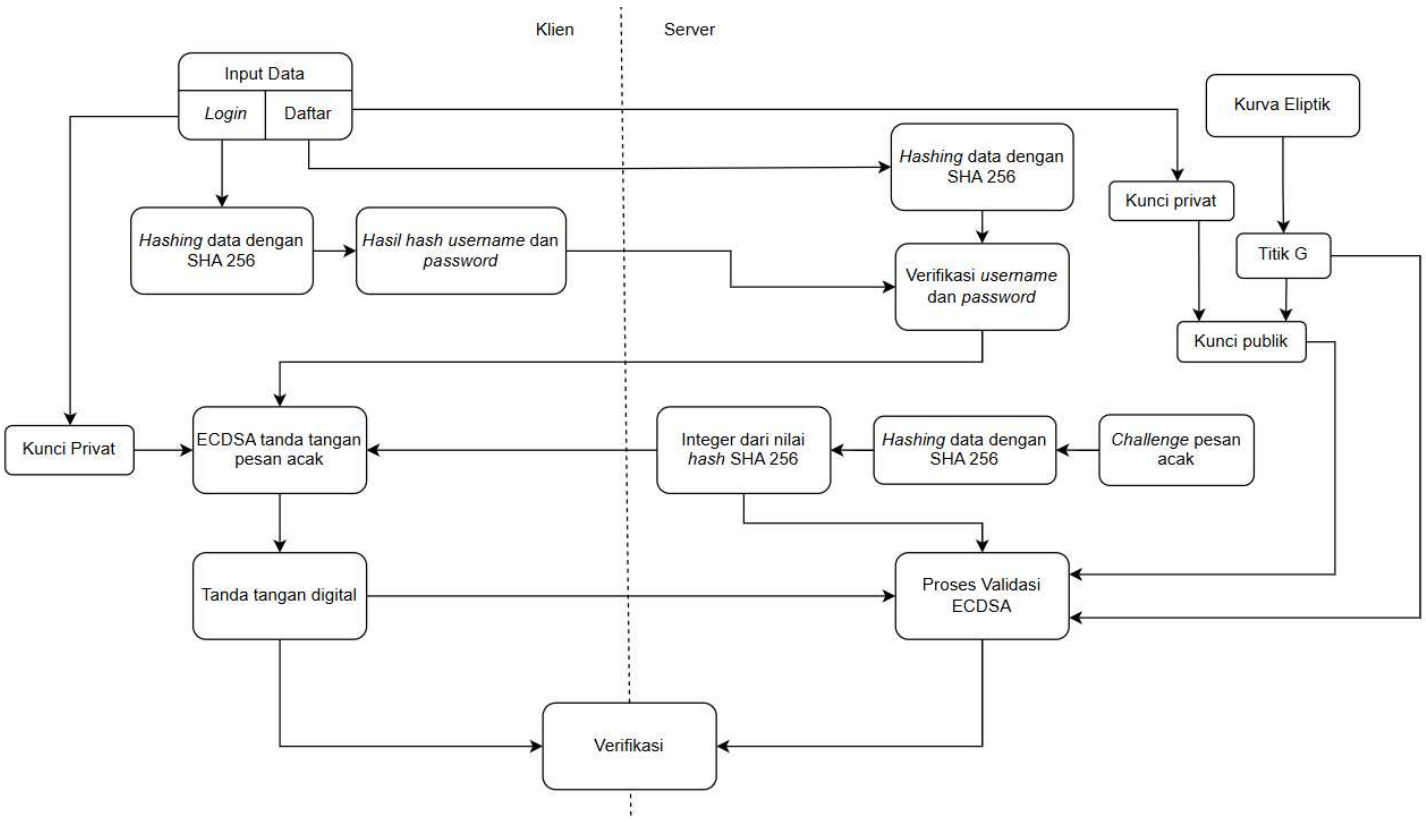
Sistem 2FA semakin sering digunakan sebagai metode keamanan dan identifikasi pengguna yang lebih aman. 2FA yang diimplementasikan dalam penelitian ini bertujuan untuk meningkatkan keamanan sistem dengan menambahkan lapisan autentikasi kedua. Hal ini membantu mengurangi risiko pengguna yang tidak sah atau berbahaya mengakses akun *client* ketika login. Bagan alur 2FA dapat dilihat pada Gambar 3.3.



Gambar 3.3 Flowchart 2FA

3.3 Pengembangan Model

Pada bagian ini akan ditunjukkan skema program autentikasi login akun dengan 2FA yang dikombinasikan dengan ECDSA dan SHA 256. Skema program terdiri dari beberapa tahap, yaitu pendaftaran akun, login akun, dan verifikasi serta autentikasi saat login akun. Skema pengembangan model dapat dilihat pada Gambar 3.4.



Gambar 3.4 Skema Pengembangan Model

3.4 Konstruksi Program

3.4.1 *Input* dan *Output*

Pada bagian ini akan disampaikan rancangan dari *input* yang diberian serta *output* dari program pada proses pendaftaran akun, login akun, dan verifikasi serta autentikasi pada saat melakukan login akun. *Input* dan *output* yang akan digunakan pada penelitian ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 *Input* dan *output* program

	Pendaftaran Akun	Login Akun	Verifikasi dan Autentikasi
Input	• *Username* dan *password* akun yang akan didaftarkan • Kunci privat *client*	• *Username* dan *password client* yang telah didaftarkan • *File txt* berisi *username* dan *password*	• *File txt* berisi kunci privat *client* • Kunci publik

Tabel 3.2 *Input* dan *output* program

	Pendaftaran Akun	Login Akun	Verifikasi dan Autentikasi
Output	• *File txt* berisi *username* dan *password* • *File txt* berisi kunci privat *client* • *File txt* berisi kunci privat dalam bentuk fungsi *hash* dengan SHA 256	• Keterangan apakah *username* dan *password* terdaftar dan valid atau tidak	• Keterangan apakah kunci privat valid atau tidak

3.4.2 Algoritma Deskriptif

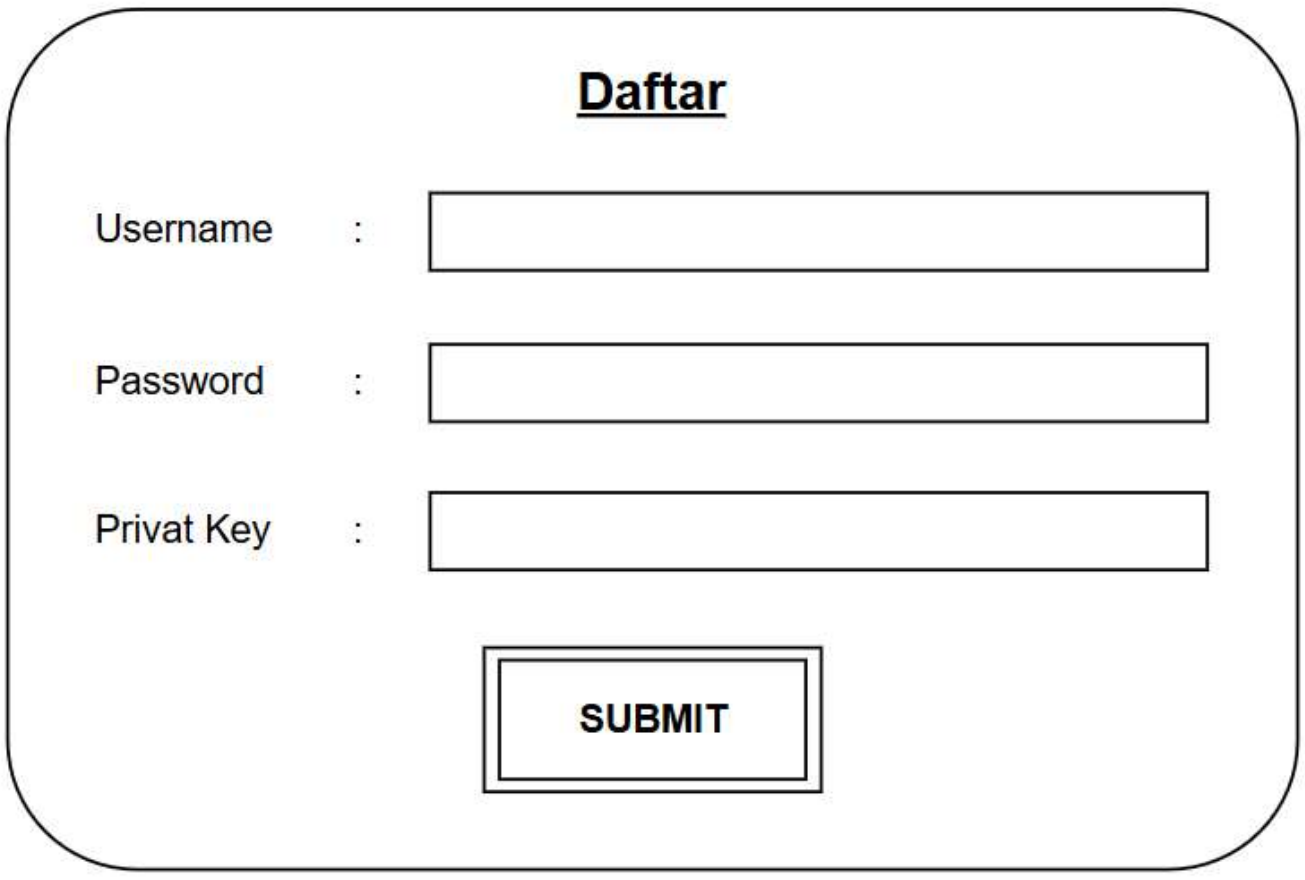
Algoritma deskriptif untuk *client* dan *server* adalah sebagai berikut:

a. Pendaftaran akun *client*
   1. Masukkan *username*, *password*, dan kunci privat.
   2. Tekan tombol "SUBMIT".
   3. Diperoleh akun dan *file txt* berisi kunci privat yang telah di-*hash*.

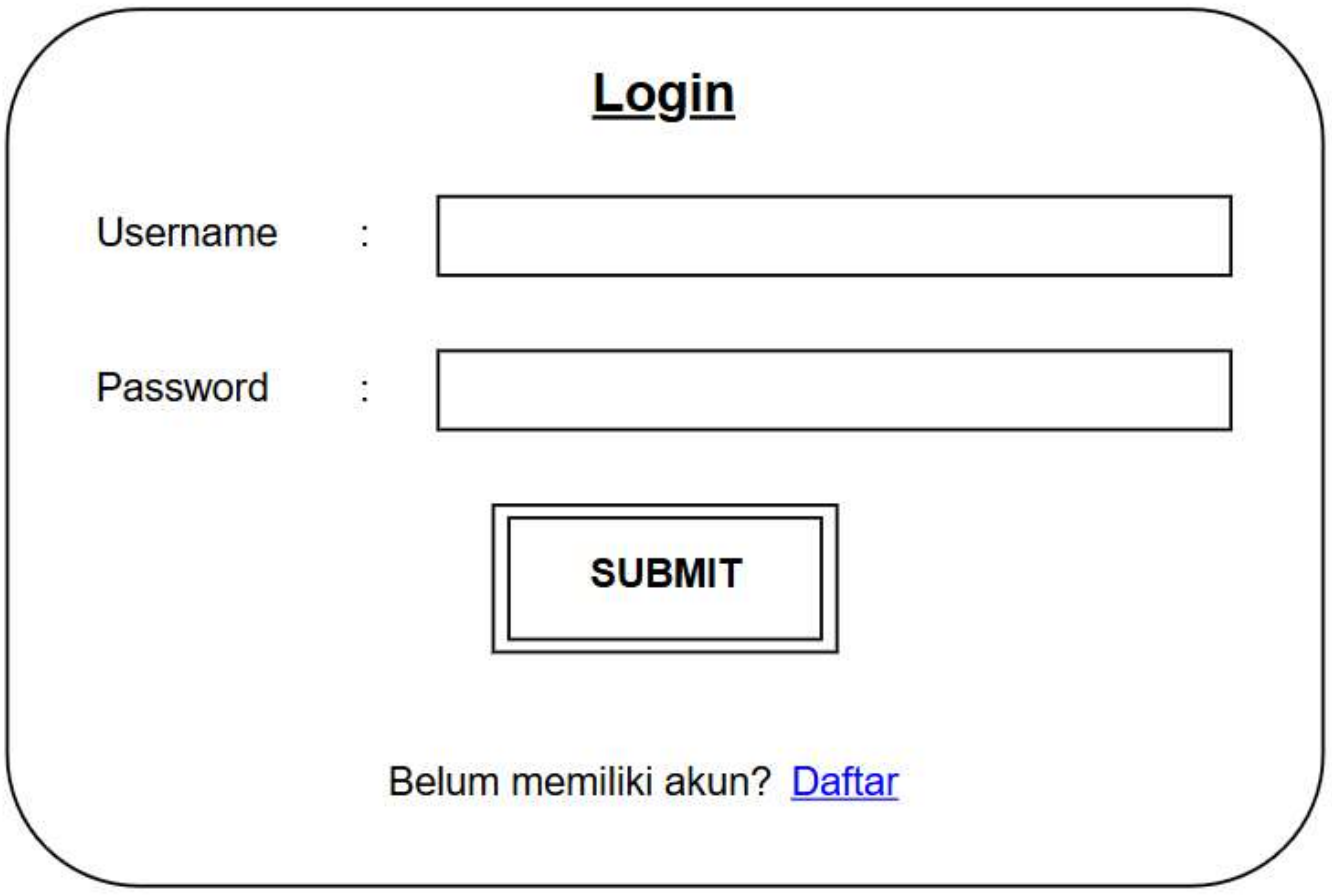
b. Login akun *client*
   1. Masukkan *username* dan *password*.
   2. Tekan tombol "SUBMIT".
   3. Tanda tangan pesan acak (secara otomatis).
   4. Tunggu verifikasi.

c. *Server*
   1. Melakukan verifikasi *hash* SHA 256 dari *username* dan *password*.
   2. Kirimkan *challenge* berupa pesan acak ke *client*.
   3. Melakukan verifikasi tanda tangan ECDSA.

3.4.3 Rancangan Tampilan



Gambar 3.5 Tampilan Halaman Pendaftaran Akun



Gambar 3.6 Tampilan Halaman Login Akun

Gambar 3.7 Tampilan Setelah Memasukkan Username dan Password Benar

Gambar 3.8 Tampilan Login Berhasil

Gambar 3.9 Tampilan Login Gagal Karena Username / Password Salah

Gambar 3.10 Tampilan Login Gagal Karena Kunci Privat Tidak Valid

3.5 *Library* Python

1. Tkinter

Tkinter adalah *library* yang dapat digunakan untuk membuat antarmuka sistem operasi yang berbasis grafis. Tkinter dapat memudahkan penggunaan aplikasi yang dibuat dengan tampilan yang lebih *user-friendly*. *Library* ini

mempermudah dalam pembuatan aplikasi dengan berbagai komponen seperti tombol, kolom pengisian, dan lainnya.

2. Hashlib

   Hashlib merupakan *library* yang memiliki fungsi untuk melakukan *hashing* dengan berbagai algoritma seperti SHA-1, SHA-256, SHA-512, dan MD5. Pada penelitian ini, hashlib digunakan untuk melakukan *hashing* dengan SHA-256.

3. Os

   Os adalah *library* yang digunakan untuk operasi terkait *file* (seperti memastikan *file* privat tersedia di perangkat *client*).

4. Random

   Random adalah *library* yang digunakan untuk menghasilkan keluaran acak untuk digunakan pada saat *server* mengirimkan *challenge* berupa pesan acak.

3.6 Proses Validasi

Pada tahapan ini akan dilakukan implementasi program yang telah dikonstruksi. Pada penelitian ini, validasi dilakukan dengan cara mendaftarkan akun, kemudian akan dilakukan percobaan dengan empat kasus yang berbeda sebagai berikut:

1. Melakukan login akun dengan memasukkan *username* dan *password* yang benar, kemudian menandatangani pesan acak dengan menggunakan *file* kunci privat yang telah didaftarkan sebelumnya, lalu *output* akan menunjukkan tanda tangan digital autentik dan login berhasil.
2. Melakukkan login akun dengan memasukkan *username* atau *password* yang salah, lalu *output* akan menunjukkan *username* atau *password* salah dan login gagal.
3. Melakukkan login dengan memasukkan *username* dan *password* yang benar, kemudian menandatangi pesan acak dengan menggunakan *file* kunci privat yang telah diubah isi *hash*-nya, lalu *output* akan menunjukkan tanda tangan digital tidak autentik dan login gagal.
4. Menghapus *file* kunci privat yang telah didaftarkan sebelumnya, kemudian melakukkan login dengan memasukkan *username* dan *password* yang benar, lalu *output* akan menunjukkan tanda tangan digital tidak autentik dan login gagal.

3.7 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah prototipe yang telah dibuat diuji dan hasil validasinya dianalisis. Apabila prototipe berhasil divalidasi dan sesuai dengan tujuan, maka dapat disimpulkan bahwa program berfungsi dengan baik. Prototipe yang telah berfungsi dengan baik ini dapat diimplementasikan atau dijadikan dasar untuk penelitian lebih lanjut guna pengembangan di masa depan.