***DIGITAL SIGNATURE* MENGGUNAKAN ALGORITMA KRIPTOGRAFI RIVEST SHAMIR ADLEMAN (RSA) UNTUK PENGAMANAN DATA PADA SEKOLAH**

**SKRIPSI**

diajukan untuk menempuh ujian sarjana

pada Fakultas Matematika dan Ilmu Pengetahuan Alam  
Universitas Padjadjaran

SHANIA SALSABILA

NPM 140810180014



UNIVERSITAS PADJADJARAN  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
PROGRAM STUDI TEKNIK INFORMATIKA  
SUMEDANG  
2020

**SKRIPSI**

***DIGITAL SIGNATURE* MENGGUNAKAN KRIPTOGRAFI ALGORITMA RIVEST SHAMIR ADLEMAN (RSA) UNTUK PENGAMANAN DATA PADA SEKOLAH**

**DIGITAL SIGNATURE USING RIVEST SHAMIR ADLEMAN (RSA) CRYPTOGRAPHY ALGORITHM FOR DATA SECURITY AT SCHOOLS**

Telah dipersiapkan dan disusun oleh

SHANIA SALSABILA

NPM 140810180014

Telah dipertahankan di depan Tim Penguji

pada tanggal …………….

Susunan Tim Penguji

1. Dr. Juli Rejito, M.Kom. Ketua Tim Penguji

NIP. 19680717 199303 1 003 ……………………….

1. Akik Hidayat, Drs. M.Kom Pembimbing

NIP. 19611018 198603 1 002 ……………………….

1. Deni Setiana, S.Si. M.Cs. Co-Pembimbing

NIP. 19730925 200312 1 003 ……………………….

1. ……………………………… Penguji

NIP. ………………………... ……………………….

1. ……………………………… Penguji

NIP. ………………………... ……………………….

1. ……………………………… Penguji

NIP. ………………………... ……………………….

KATA PENGANTAR

*Bismillahirahmanirrahim*. Dengan menyebut nama Allah yang Maha Pengasih dan Maha Penyayang. Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul “***DIGITAL SIGNATURE* MENGGUNAKAN KRIPTOGRAFI ALGORITMA RIVEST SHAMIR ADLEMAN (RSA) UNTUK PENGAMANAN DATA PADA SEKOLAH**” sebagai salah satu syarat menempuh sarjana pada Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Dalam proses penyusunan dan penulisan skripsi ini tidak terlepas dari bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis mengucapkan terima kasih kepada Bapak Akik Hidayat, Drs. M.Kom, sebagai pembimbing utama, Bapak Deni Setiana, S.Si. M.Cs, sebagai pembimbing pendamping yang telah meluangkan waktu dan pikirannya sehingga penulis dapat menyelesaikan skripsi ini. Ucapan terima kasih juga diberikan kepada keluarga penulis yang selalu memberikan motivasi dan doa yang menjadi pendorong dalam penyelesaian skripsi ini. Penulis juga mengucapkan terima kasih sebanyak-banyaknya kepada:

Bapak Prof. Dr. Iman Rahayu, S.Si., M.Si, selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Bapak Dr. Setiawan Hadi, M.Sc.Cs, selaku Kepala Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Bapak Dr. Juli Rejito, M.Kom, selaku Ketua Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran

Bapak Akik Hidayat, Drs. M.Kom, selaku Dosen Pembimbing Utama dan Bapak Deni Setiana, S.Si. M.Cs, selaku Co. Pembimbing.

Dosen-dosen Teknik Informatika Unpad yang telah mengajar dan memberikan ilmu kepada penulis selama masa perkuliahan yang membawa penulis pada posisi sekarang ini.

Seluruh Staf Pengajar dan Tata Usaha Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam yang sudah membantu proses pembelajaran di perkuliahan selama ini.

Kedua Orangtua tercinta, Papa dan Mama yang telah memberikan cinta dan kasih sayang serta motivasi dan doa yang tidak putus kepada penulis agar bisa menyelesaikan skripsi ini. Tidak lupa juga saudara dan keluarga penulis yang telah memberikan semangat dan doa.

Kepada Wildan yang selalu menemani, memberikan motivasi agar tetap semangat mengerjakan skripsi dan juga menenangkan disaat penulis sedang *down*.

Kepada Fany dan Sifa yang sangat membantu dan menemani penulis selama berada di Jatinangor sehingga penulis tetap semangat dan terhibur disaat jenuh.

Kepada Sahabat Olimki: Indah, Mega, Alwan, Fatih, serta Bang Arief, Fikra, dan teman-teman lain yang nama nya tidak dapat penulis sebutkan satu per satu yang telah memberikan semangat dan membantu dalam penyusunan skripsi ini baik secara langsung maupun tidak.

Kepada teman-teman seperjuangan Mahasiswa Teknik Informatika angkatan 2018 yang telah memberikan pengalaman dan kenangan selama perkulihan.

Semoga Allah SWT membalas segala kebaikan yang telah mereka berikan kepada penulis. Penulis menyadari bahwa skripsi ini masih memiliki kekurangan karena terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan saran dan kritik yang membangun dari berbagai pihak. Akhir kata, penulis berharap semoga skripsi ini dapat bermanfaat bagi pembaca.

Jatinangor, 2022

Penulis

Shania Salsabila

ABSTRAK

Di era digital ini, data adalah unsur terpenting yang digunakan dalam berbagai bidang kehidupan. Semakin berkembangnya teknologi, maka semakin banyak juga orang yang merusak, meretas, dan merusak data yang telah disimpan dengan baik. Kriptografi merupakan ilmu untuk menjaga kerahasiaan data dengan menggunakan kunci. Penandatanganan secara digital dianggap dapat menjaga keamanan data juga mencegah pemalsuan. Algoritma kriptografi yang memiliki tingkat keamanan tinggi adalah kriptografi RSA (Rivest Shamir Adleman) yang juga digunakan sebagai algoritma tanda tangan digital.

Pada penelitian ini, diimplementasikan digital signature menggunakan kriptografi RSA (Rivest Shamir Adleman) untuk melakukan pengamanan data yang ada pada instansi pendidikan atau sekolah. Aplikasi yang akan dibangun akan melakukan perhitungan kriptografi RSA (Rivest Shamir Adleman), lalu mencari intisari dari sebuah data untuk digabungkan dengan digital signature. Pengujian pada penelitian ini dilakukan dengan Blackbox Testing dan hasil pengujian tersebut adalah implementasi yang berjalan dengan baik.

**Kata Kunci** : Data*,* Kriptografi, RSA (Rivest Shamir Adleman), Tanda tangan digital, Keamanan

*ABSTRACT*

*In this digital era, data is the most important element used in various fields of life. As technology evolves, more and more people are tampering with, hacking, and destroying well-stored data. Cryptography is the science of keeping data confidential by using keys. Digital signing is considered capable of maintaining data security and preventing counterfeiting. One of the cryptographic algorithms that has a high level of security is RSA* (*Rivest Shamir Adleman*) *cryptography which is also used as a digital signature algorithm.*

*In this study, digital signatures were implemented using RSA* (*Rivest Shamir Adleman*) *cryptography to secure data at an educational institution or schools. The application that will be built will perform RSA* (*Rivest Shamir Adleman*) *cryptographic calculations, then look for the essence of a data to be combined with a digital signature. Tests in this study were carried out with Blackbox Testing and the results of these tests were implementations that went well.*

***Keywords :*** *Data, Cryptography, RSA* (*Rivest Shamir Adleman*)*, Digital signature, Security*

DAFTAR ISI

[KATA PENGANTAR iii](#_Toc131503179)

[ABSTRAK v](#_Toc131503180)

[*ABSTRACT* vi](#_Toc131503181)

[DAFTAR ISI vii](#_Toc131503182)

[DAFTAR TABEL x](#_Toc131503183)

[DAFTAR GAMBAR xi](#_Toc131503184)

[DAFTAR LAMPIRAN xiii](#_Toc131503185)

[BAB I PENDAHULUAN 1](#_Toc131503186)

[1.1 Latar Belakang 1](#_Toc131503187)

[1.1. Identifikasi Masalah 3](#_Toc131503188)

[1.2 Batasan Masalah 3](#_Toc131503189)

[1.3 Maksud dan Tujuan Penelitian 4](#_Toc131503190)

[1.4 Manfaat Penelitian 4](#_Toc131503191)

[1.5 Metodologi Penelitian 4](#_Toc131503192)

[1.6 Sistematika Penulisan 5](#_Toc131503193)

[BAB II TINJAUAN PUSTAKA 7](#_Toc131503194)

[2.1 Kriptografi 7](#_Toc131503195)

[2.2 Kriptografi RSA (Rivest Shamir Adleman) 8](#_Toc131503196)

[2.3 *Hash* 10](#_Toc131503197)

[2.4 Tanda Tangan Digital (*Digital Signature)* 13](#_Toc131503198)

[2.5 Algoritma RSA untuk Tanda Tangan Digital 14](#_Toc131503199)

[2.6 Keamanan Data 15](#_Toc131503200)

[2.7 *Hacker* 17](#_Toc131503201)

[2.8 Java 20](#_Toc131503202)

[2.9 *Unified Modelling Language* (UML) 21](#_Toc131503203)

[2.9.1 *Use Case Diagram* 21](#_Toc131503204)

[2.9.2 *Activity Diagram* 22](#_Toc131503205)

[2.9.3 *Sequence Diagram* 23](#_Toc131503206)

[2.10 *Blackbox Testing* 24](#_Toc131503207)

[BAB III ANALISIS DAN PERANCANGAN 25](#_Toc131503208)

[3.1 Kerangka Kerja Penelitian 25](#_Toc131503209)

[3.1.1 Deskripsi Umum Sistem 25](#_Toc131503210)

[3.1.2 Analisis Metode RSA (Rivest Shamir Adleman) 25](#_Toc131503211)

[3.1.3 Analisis Pembentukan Message Digest 25](#_Toc131503212)

[3.1.4 Analisis Penggabungan Pesan dan Digital Signature 26](#_Toc131503213)

[3.2 Analisis Kebutuhan 26](#_Toc131503214)

[3.2.1 Kebutuhan Perangkat Keras 26](#_Toc131503215)

[3.2.2 Kebutuhan Perangkat Lunak 26](#_Toc131503216)

[3.2.3 Kebutuhan Data 27](#_Toc131503217)

[3.2.4 Prosedur 27](#_Toc131503218)

[3.3 Perancangan Aplikasi 27](#_Toc131503219)

[3.3.1 Sistem RSA (Rivest Shamir Adleman) *Digital Signature* 27](#_Toc131503220)

[3.3.2 Flowchat Proses Digital Signature 29](#_Toc131503221)

[3.3.3 Use *Case diagram* 30](#_Toc131503222)

[3.3.4 Activity *diagram* 31](#_Toc131503223)

[3.3.5 Sequence Diagram 32](#_Toc131503224)

[3.4 Desain Tampilan Program 33](#_Toc131503225)

[3.5 *Blackbox Testing* 35](#_Toc131503226)

[BAB IV HASIL DAN PEMBAHASAN 37](#_Toc131503227)

[4.1 Impelementasi Program 37](#_Toc131503228)

[4.1.1 Halaman Pertama 37](#_Toc131503229)

[4.1.2 Halaman Kedua 39](#_Toc131503230)

[4.1.3 Halaman Ketiga 41](#_Toc131503231)

[4.2 Pengujian 43](#_Toc131503232)

[BAB V KESIMPULAN DAN SARAN 44](#_Toc131503233)

[5.1 Kesimpulan 44](#_Toc131503234)

[5.2 Saran 44](#_Toc131503235)

[DAFTAR PUSTAKA 45](#_Toc131503236)

[LAMPIRAN 47](#_Toc131503237)

[RIWAYAT HIDUP 61](#_Toc131503238)

DAFTAR TABEL

[Tabel 2.1 Simbol-simbol pada *Use Case Diagram* 14](#_Toc130900068)

[Tabel 2.2 Simbol-simbol pada *Activity Diagram* 15](#_Toc130900069)

[Tabel 2.3 Simbol-simbol pada Sequence Diagram 16](#_Toc130900070)

DAFTAR GAMBAR

[Gambar 2.1 Penulisan Rahasia (Denning, 1992) 7](#_Toc128430053)

[Gambar 2.2 Skema tanda tangan digital 10](#_Toc128430054)

[Gambar 2.3 Sistem RSA Digital Signature 10](#_Toc128430055)

[Gambar 3.1 Flowchart proses digital signature 17](#_Toc117195077)

[Gambar 3.2 *Use Case Diagram* 17](#_Toc117195078)

[Gambar 3.3 Avtivity Diagram 18](#_Toc117195079)

[Gambar 3.4 *Sequence Diagram* 19](#_Toc117195080)

[Gambar 3.5 Frame 22](#_Toc117195081)

[Gambar 3.6 Frame 2 22](#_Toc117195082)

[Gambar 3.7 Frame 3 23](#_Toc117195083)

[Gambar 4.1 Halaman Pertama 25](#_Toc117195084)

[Gambar 4.2 Halaman Kedua 28](#_Toc117195085)

[Gambar 4.3 Halaman Ketiga 30](#_Toc117195086)

# **DAFTAR LAMPIRAN**

[Lampiran 1 Ringkasan Hasil *Usability Testing* 34](#_Toc35650951)

[Lampiran 2 Kode Web Aplikasi 34](#_Toc35650952)

# **BAB I**

**PENDAHULUAN**

1. **Latar Belakang**

Data atau informasi adalah unsur terpenting yang digunakan dalam berbagai bidang kehidupan dan sebagai acuan atau dasar penelitian dalam setiap implementasi suatu kegiatan. Data dapat dikirimkan secara *offline* maupun *online*. Pengiriman *online* mudah dilakukan dan sangat efektif ketika jarak menjadi kendala. Seiring perkembangan teknologi dan internet, maka ancaman dan kerentanan informasi meningkat dengan munculnya peretasan/kebocoran informasi juga semakin banyak orang yang mampu merusak data meskipun telah disimpan dengan hati-hati. Disini dibutuhkan jaminan bahwa data terjaga dari berbagai ancaman dan tidak mengalami perubahan saat dikirimkan.

Keamanan informasi data adalah upaya untuk melindungi data agar tidak diakses, digunakan, diganggu, diubah, atau dimusnahkan oleh pihak yang tidak bertanggung jawab dalam menjaga integritas dan kerahasiaan data. Upaya pengamanan data dikembangkan dalam berbagai teknik, seperti penggunaan kriptografi yang saat ini sedang berkembang. Kriptografi adalah ilmu yang menggunakan persamaan matematika untuk mengenkripsi data dengan tujuan mengamankan isi data atau menjaga kerahasiaan informasi dari orang-orang yang tidak berhak. Dalam hal ini data akan ditandatangani secara digital sehingga dapat terjaga keamanan dan keasliannya. Penandatanganan secara digital atau *digital signature* dianggap dapat menjamin keamanan dokumen juga dapat mencegah pemalsuan dokumen.

Tanda tangan digital dihasilkan dari data/*file* selama proses perhitungan matematis yang kompleks. Setiap jenis *file* memiliki tanda tangan digital yang berbeda. Pada dasarnya, tanda tangan digital terdiri dari dua proses, yaitu proses penandatanganan dan proses verifikasi. Tanda tangan digital dapat menjamin beberapa aspek keamanan, seperti *authencity*, *integrity*, dan *non-repudiation*. *Authenticy* memberi tahu siapa yang membuat dokumen. *Integrity* berarti tidak ada perubahan yang dilakukan pada dokumen setelah tanda tangan diberikan. *Non-repudiation* artinya penulis dokumen tidak dapat menyangkal bahwa dia membuat dokumen tersebut di kemudian hari (Azdy, 2016).

Sebagian besar skema tanda tangan digital digunakan oleh algoritma enkripsi asimetris, yang melibatkan RSA (Rivest Shamir Adleman), DSA (*Digital Signature Algorithm*), dan ECDSA (*Elliptic Curve Digital Signature Algorithm*). Secara umum DSA (*Digital Signature Algorithm*) hanya dapat digunakan sebagai tanda tangan digital, bukan enkripsi. Khusus RSA (Rivest Shamir Adleman) dapat digunakan sebagai algoritma tanda tangan digital atau sebagai algoritma enkripsi. RSA (Rivest Shamir Adleman) merupakan algoritma yang paling maju dalam bidang kriptografi *public key*. Mekanisme kerja algoritma ini cukup sederhana, mudah dimengerti, dan sangat kuat. Enkripsi dan dekripsinya identik, sehingga proses verifikasi dan tanda tangan juga identik. Algoritma ini didasarkan pada proses matematis yang menciptakan kunci rahasia yang dapat dibagikan. Dasar dari algoritma ini adalah perhitungan bilangan besar sebagai faktor prima (Kridalaksana et al., 2017).

Berdasarkan latar belakang yang telah dipaparkan, penulis mengusulkan aplikasi yang dapat digunakan untuk mengamankan data menggunakan kriptografi algoritma Rivest Shamir Adleman (RSA).

* 1. **Identifikasi Masalah**

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, masalah yang akan dicari solusinya dalam penelitian ini adalah sebagai berikut:

Dalam hal apakah *digital signature* dapat membantu melakukan pengamanan data?

Bagaimana konsep algoritma kriptografi RSA (Rivest Shamir Adleman) dapat digunakan untuk pengamanan data?

Sejauh mana tingkat keamanan dari penggunaan *digital signature*?

1. **Batasan Masalah**

Dari identifikasi masalah tersebut, maka dalam penelitian ini terdapat beberapa batasan masalah pada sebagai berikut:

1. Aplikasi dibangun dengan menggunakan algoritma RSA (Rivest Shamir Adleman)*.*
2. Aplikasi yang dibangun mengimplementasikan *digital signature*.
3. Aplikasi menggunakan bahasa pemrograman Java.
4. Aplikasi dibangun menggunakan framework Netbeans dan tidak terlalu fokus pada tampilan.
5. Jenis data yang akan diamankan berupa file pdf, doc, excel, dan txt.
6. **Maksud dan Tujuan Penelitian**

Maksud dari penelitian ini adalah melihat hasil dari implementasi *digital signature* menggunakan algoritma kriptografi RSA (Rivest Shamir Adleman) untuk mengamankan data.

Tujuan yang ingin dicapai oleh penulis dari penelitian ini adalah:

1. Dapat mengimplementasikan *digital signature* untuk melakukan pengamanan data.
2. Dapat mengetahui tentang konsep algoritma kriptografi RSA (Rivest Shamir Adleman) pada pengamanan data.
3. Dapat mengetahui tingkat keamanan dari penggunaan *digital signature.*
4. **Manfaat Penelitian**

Manfaat yang diharapkan dari penelitian ini adalah:

1. Membantu melakukan pengamanan data.
2. Mengatasi masalah ancaman pembajakan data atau dokumen.
3. Dapat menambah pengetahuan mengenai kriptografi RSA (Rivest Shamir Adleman) pada pengamanan data*.*
4. **Metodologi Penelitian**

Jenis penelitian yang akan digunakan melalui tahapan sebagai berikut:

1. Studi literatur. Pada tahap ini penulis mencari dan mempelajari referensi dan informasi yang berhubungan dengan penelitian dari berbagai sumber seperti buku, jurnal, maupun artikel yang mendukung penelitian.
2. Penerapan aplikasi. Pada tahap ini dilakukan perancangan aplikasi kriptografi RSA (Rivest Shamir Adleman).
3. Penulisan laporan. Pada tahap ini penulis menuliskan hasil penelitian ke dalam laporan skripsi.
4. **Sistematika Penulisan**

Untuk memberi gambaran yang jelas tentang penelitian ini, maka disusunlah sistematika penulisan yang berisi materi yang akan dibahas pada setiap bab. Sistematika dalam penulisan tugas akhir ini adalah sebagai berikut:

**BAB I PENDAHULUAN**

Pada bab ini dijelaskan tentang latar belakang dari topik penulisan skripsi, pokok permasalahan berupa identifikasi dan batasan masalah, tujuan dan manfaat yang diharapkan dari penulisan skripsi, metodologi yang digunakan serta sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Pada bab ini dijelaskan seluruh landasan teori yang berhubungan dengan penelitian, yaitu tentang metode pengembangan perangkat lunak yang digunakan, penjelasan teoritis mengenai bahasa pemrograman dan *framework* yang digunakan dalam proses pengimplementasian aplikasi, serta teori lainnya guna memahami permasalahan yang dibahas.

**BAB III ANALISIS DAN PERANCANGAN**

Pada bab ini dijelaskan tentang metode pengembangan aplikasi yang digunakan meliputi analisis kebutuhan sistem, perancangan aplikasi, diagram pemodelan sistem, model perancangan data dan rancangan antarmuka pengguna.

**BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini dijelaskan tentang implementasi aplikasi yang telah dibangun, tampilan aplikasi, pengujian aplikasi, serta hasil dari penggunaan algoritma RSA (Rivest Shamir Adleman).

**BAB V KESIMPULAN DAN SARAN**

Bab ini merupakan penutup yang berisi kesimpulan dan saran dari penelitian yang sudah dilakukan.

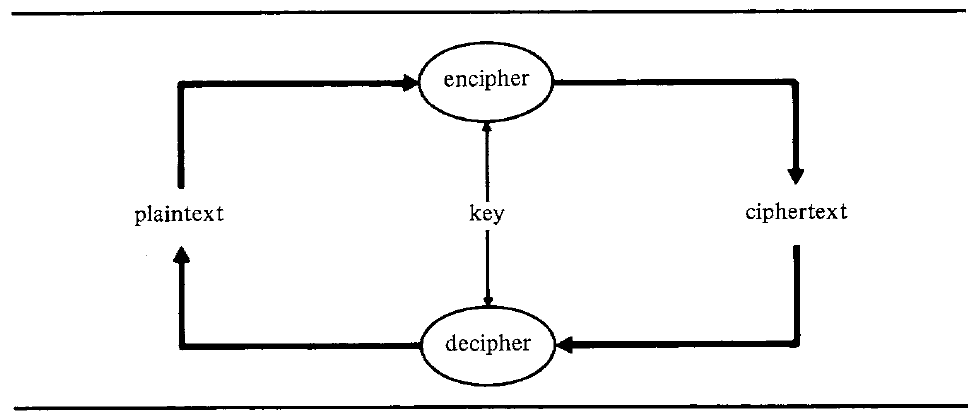
# **BAB II**

**TINJAUAN PUSTAKA**

* 1. **Kriptografi**

Kriptografi (*cryptography*) berasal dari Bahasa Yunani, yaitu *cryptos* dan *graphein*. *Cryptos* berarti rahasia, sedangkan *graphein* memiliki arti menulis. Secara umum kriptografi diartikan sebagai seni penyandian untuk menjaga keamanan juga kerahasiaan pesan. Kriptografi merupakan ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi (Munir, R, 2006).

Kriptografi adalah ilmu yang mempelajari cara mengamankan data saat dikirim dari pengirim ke penerima tanpa gangguan pihak ketiga. Kriptografi terdiri dari proses enkripsi yaitu proses melakukan perubahan kode menjadi tidak bisa dimengerti, dan proses dekripsi yaitu proses mengembalikan informasi ke bentuk aslinya. Proses ini dikendalikan oleh kunci seperti yang ditunjukkan Gambar 2.1



Gambar 2. Penulisan Rahasia (Denning, 1992)

Kriptografi digunakan untuk memecahkan masalah keamanan seperti privasi dan otentikasi. Sistem privasi mencegah ekstraksi informasi oleh pihak yang tidak berwenang dari pesan yang dikirimkan, sehingga memastikan pengirim pesan hanya akan dibaca oleh penerima yang dimaksud. Sistem otentikasi mencegah penyisipan pesan yang tidak sah, sehingga menjamin keabsahan pesan bagi penerima pesan.

* 1. **Kriptografi RSA (Rivest Shamir Adleman)**

Algoritma RSA (Rivest Shamir Adleman) diciptakan oleh tiga orang peneliti dari MIT (Massachussets Institute of Technology) tahun 1976, yaitu Ron Rivest, Adi Shamir dan Leonard Adleman. RSA (Rivest Shamir Adleman) adalah contoh pertama dan paling terkenal dari kriptosistem kunci publik (Diffie & Hellman, 1976). Keamanan RSA (Rivest Shamir Adleman) didasarkan pada kesulitan memfaktorkan bilangan bulat besar.

RSA (Rivest Shamir Adleman) merupakan teknik kriptografi dimana kunci untuk melakukan enkripsi berbeda dengan kunci dekripsi. Kunci enkripsi disebut kunci publik, sedangkan kunci dekripsi disebut kunci privat. Orang dengan kunci publik dapat mengenkripsi, tetapi hanya yang memiliki kunci privat yang dapat mendekripsi. Kunci publik dapat dimiliki orang siapa saja, tapi kunci privat hanya dimiliki oleh orang tertentu.

Berikut besaran yang digunakan dalam algoritma kriptografi RSA (Rivest Shamir Adleman):

p dan q adalah bilangan prima

n = p x q

π(n) = (p – 1)(q – 1)

e = kunci enkripsi

d = kunci dekripsi

m = plaintext

c = ciphertext

Langkah-langkah pembentukan kunci pada algoritma RSA (Rivest Shamir Adleman) untuk mendapatkan kunci publik dan kunci privat:

1. Pilih dua bilangan prima besar dan berbeda, misalkan p dan q dimana p ≠ q dan keduanya memiliki panjang bit yang sama.
2. Hitung nilai n = p . q, disebut modulus. Nilai n ini digunakan untuk proses enkripsi dan dekripsi.
3. Hitung nilai (n) = (p – 1) (q – 1), disebut phi. Nilai phi digunakan untuk memilih bilangan e dan menghitung nilai d.
4. Pilih bilangan bulat e yang relatif prima terhadap π(n) dan lebih kecil dari π(n). Nilai e ini akan menjadi kunci publik.
5. Hitung nilai d, dimana e . d = 1 mod π(n), artinya e dan d saling invers modular. Nilai d ini akan menjadi kunci privat.
6. Untuk mengenkripsi pesan, konversikan pesan ke dalam bentuk bilangan bulat m dimana 0 ≤ m < n. Lalu hitung nilai c = me mod n. Nilai c adalah hasil enkripsi.
7. Untuk mendekripsi pesan, hitung nilai m = cd mod n. Kemudian konversi kembali nilai m menjadi bentuk pesan semula.
   1. ***Hash***

Fungsi *hash* adalah fungsi matematika yang mengkompresi masukan dengan panjang sembarang ke hasil dengan panjang tetap. Kemudian kode *hash* ditambahkan ke informasi. Otentikasi dalam *hash* tergantung kepada kerahasiaan jumlah data besar yang didasarkan pada kerahasiaan dan keaslian kunci pendek. Fungsi *hash* merupakan alat yang kuat dalam desain teknik untuk melindungi keaslian informasi. Fungsi *hash* digunakan untuk mengalokasikan penyimpanan yang sama untuk *record* dari sebuah *file* (Preneel, 2010).

Dalam *hash* ada istilah teknik *non-repudiasi*, dimana penerima diberikan jaminan keaslian pesan, berarti penerima selanjutnya dapat membuktikan kepada pihak ketiga bahwa pesan itu otentik bahkan jika pengirim kemudian membatalkannya. Fungsi *hash* dapat memberikan jaminan perlindungan integritas data dengan menambahkan redudansi ke informasi dengan maksud agar memungkinkan penerima untuk membuat perbedaan antara informasi asli dan palsu sehingga dapat mengetahui apakah informasi itu *valid* atau tidak.

Salah satu aplikasi yang menggunakan fungsi *hash* adalah *digital signature*. Disini fungsi *hash* digunakan untuk mengompresi informasi yang menghasilkan nilai *hash* dari pesan asli atau intisari pesan untuk kemudian ditandatangani dengan kunci pribadi pengirim. Untuk memverifikasi tanda tangan, hasil dari tanda tangan dibandingkan dengan kode *hash* yang dapat dihitung dari informasi menggunakan kunci publik pengirim.

Dalam fungsi *hash* perlu ditekankan bahwa fungsi *hash* tidak dapat digunakan untuk enkripsi, karena dua alasan mendasar. Pertama adalah fakta bahwa fungsi *hash* tidak memiliki kunci, karena fungsi *hash* tidak membutuhkan kunci rahasia untuk menghasilkan nilai *hash*. Yang kedua adalah fungsi *has*h tidak dapat dibalik karena merupakan fungsi matematika satu arah (*one-way-function*) sehingga intisari pesan tidak dapat “didekripsi” untuk menghasilkan nilai teks biasa yang unik. Fungsi *hash* hanya bisa menghasilkan nilai *hash* dari *input* pesan atau data, tetapi tidak mungkin mengembalikan pesan asli dari nilai *hash* tersebut (Stinson & Paterson, 2019).

Ada beberapa jenis fungsi *hash* yang dapat digunakan untuk *digital signature* seperti berikut:

1. MD5 (*Message Digest* 5)

MD5 adalah salah satu fungsi *hash* yang paling umum digunakan dalam *digital signature*. Fungsi *hash* ini menghasilkan nilai *hash* 128-bit yang unik dan tetap untuk setiap pesan nya. Walaupun MD5 banyak digunakan, namun keamanannya dipertanyakan karena sering terjadi kolisi nilai *hash* (nilai *hash* yang sama untuk pesan yang berbeda).

1. SHA-1 (*Secure Hash Algorithm* 1)

SHA-1 adalah fungsi *hash* lain yang sering digunakan dalam *digital signature*. Fungsi *hash* ini menghasilkan nilai *hash* 160-bit yang unik dan tetap untuk setiap pesan yang diberikan. Namun keamanan SHA-1 juga dipertanyakan karena juga sering terjadi kolisi nilai *hash*.

1. SHA-256 (*Secure Hash Algorithm* 256)

SHA-256 adalah fungsi *hash* yang lebih aman daripada MD5 dan SHA-1. Fungsi *hash* ini menghasilkan nilai *hash* 256-bit yang unik dan tetap untuk setiap pesan. Keamanan SHA-256 cukup baik dan umum digunakan dalam *digital signature*, meskipun masih ada beberapa celah keamanan yang dapat dieksploitasi.

1. SHA-512 (*Secure Hash Algorithm* 512)

SHA-512 adalah fungsi *hash* yang paling aman dalam *digital signature*. Fungsi *hash* ini menghasilkan nilai *hash* 512-bit yang unik dan tetap untuk setiap pesan nya. Keamanan SHA-512 sangat baik dan lebih sulit untuk dieksploitasi daripada SHA-256, tapi dalam proses *hash* nya memakan waktu dan sumber daya yang lebih banyak.

Secara umum, semakin besar nilai *hash* yang dihasilkan, maka semakin sulit memalsukan tanda tangan digital. Tetapi, semakin besar ukuran nilai *hash* maka semakin banyak waktu dan sumber daya yang digunakan dalam proses *hash*. Oleh karena itu, penting dilakukan pemilihan fungsi *hash* dalam *digital signature* sesuai kebutuhan yang diinginkan.

Berikut adalah langkah-langkah proses *hash* secara umum:

1. Siapkan pesan atau data yang ingin di-*hash*.

Proses hash dimulai dengan menyiapkan pesan atau data yang akan di hash, bisa berupa teks, gambar, atau file.

1. Tentukan fungsi *hash* yang akan digunakan.

Selanjutnya menentukan fungsi *hash* yang digunakan untuk menghasilkan nilai *hash* dari pesan atau data. Contoh fungsi *hash* yang biasa digunakan adalah SHA-256, MD5, dan SHA-1.

1. Ubah pesan atau data menjadi bentuk yang dapat diproses oleh fungsi *hash*.

Pesan atau data yang akan di-*hash* harus diubah menjadi bentuk yang biasanya dipecah menjadi blok-blok yang lebih kecil agar dapat diproses dengan mudah.

1. Lakukan proses *hash*

Proses *hash* dilakukan dengan menerapkan fungsi *hash* pada setiap blok pesan atau data yang telah dipecah. Setelah setiap blok diolah, nilai *hash* dari blok tersebut digabungkan dengan nilai *hash* dari blok sebelumnya untuk membentuk nilai *hash* keseluruhan

1. Hasilkan nilai *hash*

Setelah seluruh blok diolah, maka dihasilkan nilai *hash* dari pesan atau data tersebut. Nilai *hash* ini bersifat unik dan tetap untuk setiap pesan atau data yang diberikan. Nilai dari *hash* akan digunakan sebagai *input* untuk algoritma *digital signature*.

* 1. **Tanda Tangan Digital (*Digital Signature)***

Tanda tangan digital adalah teknik yang menggunakan algoritma matematika untuk memverifikasi integritas dan keaslian dokumen elektronik, email, atau perangkat lunak (Pial, Hamidur Rahaman, 2020). Tanda tangan digital digunakan untuk memastikan bahwa file yang ditransmisikan pemilik yang ditentukan dapat dituju kepada penerima dalam format aslinya tanpa adanya gangguan apapun.

Fungsi utama dari *digital signature* adalah *non-repudiation* atau anti penyangkalan, pengirim tidak bisa menyangkal bahwa dokumen *valid* dan benar dikirimkan oleh pengirim yang bersangkutan. Tanda tangan digital dapat digunakan pada pesan apa pun, terenkripsi atau tidak, sehingga penerima dapat memverifikasi identitas pengirim dan pesan yang masuk tetap utuh.

Diagram

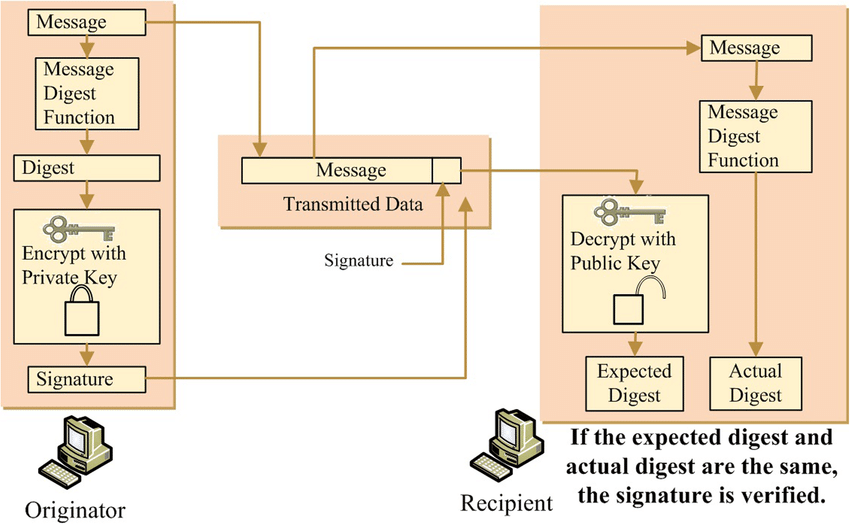
Description automatically generated

Gambar 2. Skema tanda tangan digital

Tanda tangan digital adalah alat yang penting dalam mengamankan informasi digital dan memverifikasi identitas penandatangan. Contoh penggunaannya termasuk dalam email, dokumen digital, transaksi keuangan online, dan sertifikat digital. Dalam semua contoh tersebut, tanda tangan digital digunakan untuk memastikan keaslian dan integritas informasi serta membantu memverifikasi identitas penandatangan.

* 1. **Algoritma RSA untuk Tanda Tangan Digital**

RSA (Rivest Shamir Adleman) adalah algoritma pertama yang diketahui dapat digunakan sebagai algoritma tanda tangan digital dan enkripsi. RSA (Rivest Shamir Adleman) lebih unggul dibandingkan dengan algoritma DSA, karena DSA hanya dapat digunakan untuk tandatangan digital, bukan enkripsi. RSA (Rivest Shamir Adleman) adalah algoritma yang mengimplementasikan kriptosistem kunci publik yang ide utamanya adalah enkripsi kunci publik dan tanda tangan.



Gambar 2. Sistem RSA (Rivest Shamir Adleman) Digital Signature

* 1. **Keamanan Data**

Keamanan data adalah ilmu yang mempelajari metode perlindungan data dalam komputer dan sistem komunikasi (Dorothy Elizabeth Denning, 1992). Keamanan data merupakan upaya untuk melindungi dan mengamankan tiga aspek terpenting dunia siber atau dunia internet yaitu: kerahasiaan data, keutuhan data, ketersediaan data. Kerahasiaan data memastikan bahwa pengguna internet terlindungi privasinya baik privasi komputer pribadi maupun data selama beragai aktivitas penjelajahan internet. Keutuhan data memastikan pengguna internet menerima data yang lengkap dan akurat tanpa dirubah dan dimodifikasi oleh pihak lain di tengah jalan. Ketersediaan data memastikan bahwa pengguna internet mendapatkan data mereka kapan pun mereka mau tanpa disembunyikan dan tidak terhalang oleh pihak lain (Garfinkel & Lipford, 2014).

Keamanan data adalah konsep yang mencakup berbagai tindakan untuk melindungi data dari ancaman keamanan yang berpotensi merusak, mengakses, atau mengubah data dengan tidak sah saat ditransfer melalui jaringan. Evolusi keamanan data dimulai dengan adanya kejahatan komputer. Kejahatan komputer paling umum di era modern telah tercatat sejak akhir 70-an oleh *hacker* yang bernama Kevin Mitnick (Middleton, B, 2017). Keamanan data meliputi berbagai aspek, seperti:

1. Keamanan akses: meliputi proses untuk memastikan bahwa hanya orang yang berwenang yang dapat mengakses data. Hal ini dapat dilakukan dengan mengatur akses ke data, pengelolaan sandi dan hak akses, dan verifikasi identitas pengguna.
2. Kemanan jaringan: melibatkan teknik untuk melindungi data saat ditransfer melalui jaringan, seperti teknologi enkripsi dan pengamanan *firewall*.
3. Keamanan data di repositori: meliputi proses untuk melindungi data saat disimpan di *server* atau penyimpanan *cloud*. Hal ini dapat dilakukan dengan teknik enkripsi data atau kontrol akses ke data.
4. Keamanan data pribadi: melibatkan tindakan untuk melindungi data pribadi, seperti informasi pengguna, identitas, dan data finansial.
5. Pengamanan transaksi elektronik: meliputi tindakan untuk melindungi transaksi online seperti pembayaran, transfer uang, dan pembelian secara online. Hal ini dilakukan dengan teknik enkripsi data dan proses pengesahan transaksi.
6. *Backup* dan *recovery data*: meliputi proses untuk memastikan bahwa data yang hilang atau rusak dapat dipulihkan dengan cepat dan mudah.

Dengan menerapkan tindakan keamanan data yang tepat, dapat melindungi data dari ancaman keamanan dan mengurangi risiko kerusakan, pengambilan data secara tidak sah, dan penyalahgunaan data.

* 1. ***Hacker***

Menemukan kelemahan atau kesenjangan keamanan dalam sistem komputer atau jaringan dan memanfaatkannya untuk mengakses data tanpa izin atau memodifikasi fitur sistem komputer atau jaringan dikenal sebagai *hacking*. *Hacking* adalah istilah yang digunakan untuk menggambarkan perubahan jaringan komputer, perangkat lunak, atau perangkat keras untuk mencapai tujuan yang tidak sesuai dengan tujuan pengguna. Sebagai perbandingan, itu juga disebut sebagai melanggar keamanan dan mencuri informasi pribadi atau rahasia seseorang, seperti nomor telepon, informasi kartu kredit, alamat, kata sandi perbankan online, dll.

Di media umum, seseorang yang menggunakan *bug* dan eksploitasi untuk melanggar keamanan orang lain, atau yang menggunakan keahlian profesionalnya untuk bertindak secara jahat atau merusak, disebut dengan *hacker*. Para ahli perangkat keras dan program komputer adalah *hacker*. Seorang *hacker* adalah ahli dalam bahasa kode, jaringan, keamanan, dan komputer secara umum. Dia adalah tipe orang yang suk belajar tentang berbagai teknologi, spesifikasi sistem komputer, dan meningkatkan kemampuan dan keterampilannya (Gupta & Abhineet, 2017).

Menurut cara dari beroperasi atau berdasarkan niat mereka, *hacker* dapat diklasifikasikan sebagai berikut:

1. *White Hat Hackers*

*White Hat Hackers* adalah seorang *hacker* pakar keamanan komputer yang melanggar jaringan yang dilindungi atau sistem komputer perusahaan atau kelompok untuk mengidentifikasi kerentanan dan memperbaikinya untuk meningkatkan keamanan. Sebelum *hacker* jahat menemukan sebuah perusahaan atau organisasi dan menyebabkan kerusakan apapun, *white hat hackers* menggunakan keterampilan dan pengetahuan mereka untuk melindunginya. *White hat hackers* adalah orang-orang yang berwenang di industri, meskipun teknik yang mereka gunakan identik dengan *hacker* jahat, tetapi mereka diizinkan untuk melakukannya oleh organisasi atau perusahaan yang mempekerjakan mereka. *White hat hackers* seringkali bekerja untuk perusahaan keamanan informasi atau departemen keamanan dalam organisasi.

1. Black Hat Hackers

*Black Hat Hackers* juga disebut sebagai “*Cracker*” adalah seorang ahli dalam perangkat keras dan perangkat lunak komputer yang melanggar keamanan seseorang dengan maksud mencuri atau merusak informasi penting dan rahasia, membahayakan keamanan perusahaan besar, atau menonaktifkan atau mengubah operasi situs web dan jaringan. Demi kepentingan mereka sendiri, mereka melanggar keamanan jaringan. Ini adalah orang-orang yang sering mencoba menunjukkan keahlian komputer mereka yang luas dengan melakukan berbagai kejahatan siber seperti pencurian identitas, peninpuan kartu kredit, dll.

1. *Grey Hat Hackers*

*Grey Hat Hackers* adalah *hacker* komputer atau ahli keamanan yang kadang-kadang melanggar hukum tetapi tidak memiliki tujuan jahat seperti *black hat hackers*. Istilah “*Grey Hat*” berasal dari istilah “*Black Hat*” dan“*White Hat*” karena pada sisi *black hat* *hackers* secara ilegal memanfaatkan sistem komputer atau jaringan untuk menemukan kerentanan dan inginstruksikan orang lain tentang cara melakukannya, *grey hat hackers* tidak melakukan itu. Sementara pada sisi *white hat hackers* menemukan kerentanan dalam sistem komputer atau jaringan dan tidak memberitahu siapa pun sampai itu diperbaiki. *Grey hat hackers* berada mewakili antara *black hat hackers* dan *white hat hackers*.

Berikut adalah contoh dari kegiatan *hacker* yaitu:

1. Mencuri data pribadi atau rahasia: seorang *hacker* dapat mencuri data pribadi atau rahasia dari sistem atau aplikasi komputer, seperti nomor kartu kredit, informasi bank, atau informasi rahasia milik perusahaan.
2. Merusak sistem atau aplikasi komputer: seorang *hacker* dapat merusak sistem atau aplikasi komputer dengan mengirimkan virus atau melakukan serangan DoS (*Denial of Service*) yang dapat membuat sistem tidak dapat diakses.
3. Mencoba memecahkan password: seorang *hacker* dapat mencoba memecahkan *password* untuk mengakses sistem atau aplikasi komputer yang dilindungi dengan *password* yang kuat.
4. *Social Engineering*: seorang *hacker* dapat menggunakan teknik *social engineering* untuk meminta informasi pribadi atau rahasia dari korbannya, seperti nama pengguna dan kata sandi.
5. Penetrasi sistem: seorang *hacker* dapat melakukan penetrasi sistem atau menguji tingkat keamanan dengan cara menyerang sistem tersebut dan memanfaatkan celah keamanan yang ada, kemudian mengakses informasi yang disimpan di dalam sistem.
   1. **Java**

Java adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh James Gosling di Sun Microsystems, yang kemudian diakuisisi oleh Oracle Corporation. Ini dirilis pada tahun 1995 dan Java merupakan bahasa pemrograman yang paling populer digunakan. Dikarenakan bisa dijalankan disemua platform dan pada sistem operasi apapun seperti Windows, Mac, dan Linux (Chan, 2016). Java juga menjadi primadona untuk para developer android. Ini dikarenakan library java cukup lengkap sehingga memudahkan programmer.

Java memiliki sintaks dan fitur yang mirip seperti bahasa pemrograman C dan C++. Java dirancang menjadi bahasa yang relatif mudah dipelajari dan menjadi platform independen. Untuk memahaminya, kode Java harus diubah menjadi kode mesin melalui proses kompilasi. Java mengkompilasi semua kode ke dalam bytecode terlebih dahulu. Ketika ingin dijalankan, program didalam komputer (dikenal sebagai Java Virtual Machine atau JVM) mengubah bytecode menjadi kode mesin, ini dapat dijalankan pada mesin apapun yang memiliki JVM.

Java memiliki dua paket, yaitu: Java Runtime Environment (JRE) dan Java Development Kit (JDK). JRE digunakan untuk menjalankan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Sedangkan JDK digunakan untuk mengembangkan perangkat lunak Java menggunakan beberapa alat termasuk kompiler untuk mengkompilasi kode tertulis ke dalam bytecode (javac.exe), pengarsip untuk mengemas dan mendistribusikan file Java (jar.exe) dan generator dokumentasi untuk menghasilkan dokumentasi HTML dari kode Java (javadoc.exe) (Cardle, 2017).

* 1. ***Unified Modelling Language* (UML)**

Unified Modeling Language atau UML adalah bahasa standar untuk menulis software blueprints. UML digunakan untuk memvisualisasikan, menentukan, membangun, dan mendokumentasikan sistem perangkat lunak yang intensif. UML cocok untuk pemodelan sistem mulai dari sistem informasi perusahaan hingga aplikasi berbasis web terdistribusi (Booch et al., 2005).

* + 1. ***Use Case Diagram***

Use case diagram adalah sebuah diagram untuk memodelkan tampilan use case dari suatu sistem. Sebagian besar melibatkan pemodelan konteks sistem, subsistem, atau kelas, atau pemodelan persyaratan perilaku elemen-elemen ini. Use case diagram penting untuk memvisualisasikan, menentukan, dan mendokumentasikan perilaku elemen dan juga untuk menguji sistem (Booch et al., 2005). Simbol use case diagram dijelaskan pada tabel berikut:

Tabel 2. Simbol-simbol pada *Use Case Diagram*

| **No** | **Simbol** | **Keterangan** |
| --- | --- | --- |
| 1 | *Use Case*  *https://lh4.googleusercontent.com/aGoWgN9QLWt6cYhoD6If-1pVWdIGSDthlP-JHI4l_otU89K6rVLBWwUy1SpKgWMi8hH4mOUUioPHpG1pAJWEens_kDuXbhkl-SkRL405imaUUSx9_WfNtOM03LNWzAgdoMUaGqcX* | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor |
| 2 | *Actor*  *https://lh3.googleusercontent.com/tSEAg0SgSnqtT21tCniUCUu9Z7WocIUCePUXRQsbkimjod8vkF3FvQDwet-ag9UKfGbmo8NsF1Y6QN4bXYrCJgWZfS46QhtqQfOUqfl9MHCGdD7YQ1b-MjVUq4fSWu66r0gh37-b* | Orang atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat. Biasanya dinyatakan menggunakan kata benda atau nama aktor. |
| 3 | *Association*  *https://lh4.googleusercontent.com/SViGOIxkm4T82HeVAuSiZmc780safOlfVR22PvNtnWcDiPQzc4VCwLBlYcRJdTk_eSTupCQeZTIpVJe2AqMBzkaLi3ADmPAcvI_36DfEzuuvUl7YRIChlYH9mGJ23q5XR3seySmL* | Komunikasi antara aktor dengan *use case* yang berpartisipasi pada diagram |
| 4 | *Extend*  *https://lh6.googleusercontent.com/J3JBlXN1lK5hwy3QXjzpQ3k1G4oWSpznif1uRJPeB0MEmq9lMIgVZKW4nhjF5J1QYjikKo8stGbeXPfwkLzi-8rrkyPrH2-yTSUSSf_FBtb_K_1Nm-3QLZSF2sG-i5n27iGG2pf-* | Relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri meski tanpa *use case* tambahan itu. Arah panah mengarah pada *use case* yang ditambahkan. |
| 5 | *Generalization*  *https://lh3.googleusercontent.com/XwQKVQuzFNE-E2-BKEUf4hEvU_t77aWyx_JTwycI-lJcHPdabIEBYD_LRX75KGC7nQ7y76qzoSaMQEPOKqZMqgN4ahBnOqsmq2LUCrL8tNtQYNrY8sYhtJ8KmJSQSeawCdIccmvd* | Hubungan generalisasi dan spesialisasi (umum-khusus) antar dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. |
| 6 | *Include*  *https://lh6.googleusercontent.com/Pq5A44NAZ6zjy1qTIGxJ_ndVgw42YUy2zmxCRxJP4KGR3vX2n-J0XJJBmTNFR9VT8GnlGNTWtc7Wu8zZ5aC4lX7t1maDY0rddsJnBmNCrgNzZ8q8aAbtLgfOKeifGlLUPNtSAlU7* | Relasi *use case* tambahan ke sebuah *use case,* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Arah panah *include* mengarah pada *use case* yang dibutuhkan |

* + 1. ***Activity Diagram***

Activity diagram adalah diagram alur yang menunjukkan aliran kontrol dari aktivitas ke aktivitas. Activity diagram digunakan untuk memodelkan aspek dinamis dari suatu sistem. Sebagian besar melibatkan pemodelan langkah-langkah berurutan dalam proses komputasi. Activity diagram juga digunakan untuk untuk memvisualisasikan, menentukan, membangun, dan mendokumentasikan dinamika objek (Booch et al., 2005). Simbol activity diagram dijelaskan pada tabel berikut:

Tabel 2. Simbol-simbol pada *Activity Diagram*

| **No** | **Simbol** | **Keterangan** |
| --- | --- | --- |
| 1 | *Initial State* | Menggambarkan awal dimulainya suatu aliran aktivitas |
| 2 | *Final State* | Menggambarkan berakhirnya suatu aliran aktivitas |
| 3 | *Activity* | Menggambarkan aktivitas yang dilakukan dalam suatu aliran aktivitas |
| 4 | *Decision* | Menggambarkan pilihan kondisi atau cabang-cabang aktivitas tertentu |
| 5 | *Transition* | Berguna untuk menghubungkan satu komponen dengan komponen lainnya |

* + 1. ***Sequence Diagram***

Sequence diagram adalah diagram interaksi yang menekankan urutan waktu pesan atau urutan temporal. Diagram ini menunjukkan satu set peran dan pesan yang dikirim dan diterima oleh instance yang memainkan peran tersebut. Sequence diagram digunakan untuk menggambarkan tampilan dinamis dari suatu sistem (Booch et al., 2005). Simbol sequence diagram dijelaskan pada tabel berikut:

Tabel 2. Simbol-simbol pada Sequence Diagram

| **No** | **Simbol** | **Keterangan** |
| --- | --- | --- |
| 1 | *Actor*  Man | Menggambarkan orang yang sedang berinteraksi dengan sistem |
| 2 | *Entity Class* | Menggambarkan hubungan yang akan dilakukan |
| 3 | *Boundary Class* | Menggambarkan sebuah gambaran dari form |
| 4 | *Control Class* | Menggambarkan penghubung antara boundary dengan tabel |
| 5 | *Control & Life Line* | Menggambarkan tempat mulai dan berakhirnya *message* |
| 6 | *Message* | Menggambarkan pengiriman pesan |

* 1. ***Blackbox Testing***

Proses mengevaluasi perangkat lunak untuk menemukan kesalahan dikenal sebagai pengujian (*testing*). *Testing* diperlukan untuk memastikan bahwa perangkat lunak yang dibangun telah memenuhi spesifikasi yang diinginkan. Dan untuk memastikan apakah ada masalah atau kesalahan apapun di dalam sistem. Tujuan dari *testing* untuk mengevaluasi kemampuan dan kualitas dari sebuah program (Sawant et al., 2012). Pengujian perangkat lunak mengidentifikasi kelemahan dalam kode aplikasi yang harus diperbaiki, juga mengakses fungsionalitas dan keakuratan perangkat lunak melalui analisis untuk meningkatkan kualitas *software* (Ehmer & Khan, 2012).

Salah satu teknik pengujian yang digunakan adalah *Black Box Testing*. Teknik pengujian ini membutuhkan pengetahuan dasar tentang cara aplikasi bekerja. Ia tidak memiliki hubungan apa pun dengan struktur logis internal sistem dan hanya memeriksa elemen-elemen dasar. Tidak perlu mengetahui cara menjalankan kode internal aplikasi, hanya perlu memastikan bahwa input diterima dengan benar dan output diproduksi dengan benar.

Keunggulan dari *Black Box Testing* adalah pengujian dilakukan sesuai dengan persyaratan sudut pandang *user*, pengujian dilakukan oleh pihak ketiga untuk menghindari bias *programmer*, kasus pengujian dapat menunjukkan ada atau tidaknya kelas kesalahan, tes persepsi sangat mudah, efektif untuk segmen kode yang signifikan, pengembangan kasus pengujian dapat dilakukan dengan cepat.

Kelemahan dari *Black Box Testing* adalah kasus pengujian sulit dirancang tanpa spesifikasi yang jelas, pengujian yang dilakukan terbatas pada jumlah input tertentu, peluang untuk memiliki pengulangan pengujian yang sudah dilakukan oleh *programmer*, pengujian ini tidak efisien untuk menguji segmen kode yang kompleks, hasil dari pengujian ini terkadang *overestimated*.

# **BAB III**

**ANALISIS DAN PERANCANGAN**

1. **Kerangka Kerja Penelitian**

Metodologi dalam penelitian ini menggunakan metode pengumpulan data dan pengembangan sistem. Metode pengumpulan data meliputi observasi dan studi pustaka. Objek penelitian ini adalah data-data seperti file txt yang ingin diamankan. Sedangkan dalam metode pengembangan sistem melalui tahap analisis yang dilakukan untuk membangun aplikasi pembuatan tandatangan digital.

* + 1. **Deskripsi Umum Sistem**

Proses sistematis dimana desain sistem dijelaskan, desain sistem dijabarkan sebelum kode ditulis dalam Bahasa pemrograman. Rancangan umum akan diaplikasikan untuk tujuan memberikan gambaran umum bagi pengguna tentang sistem yang dibangun.

* + 1. **Analisis Metode RSA (Rivest Shamir Adleman)**

Pada tahap ini RSA (Rivest Shamir Adleman) adalah algoritma kriptografi yang digunakan untuk menghasilkan sepasang kunci publik dan kunci privat yang digunakan dalam proses enkripsi untuk mendapatkan tanda tangan digital.

* + 1. **Analisis Pembentukan Message Digest**

Pada tahap ini sebuah pesan diambil *message digest* (intisari pesan) menggunakan fungsi algoritma hash SHA-256. Proses *hash* ini merupakan salah satu tahap dalam proses *digital signature* yang membantu memastikan integritas dokumen dan tanda tangan digital.

* + 1. **Analisis Penggabungan Pesan dan Digital Signature**

Pada tahap ini pesan yang telah melewati proses *hash* untuk mendapatkan *message digest*, hasilnya akan dienkripsi menggunakan kunci privat untuk mendapatkan tanda tangan digital. Kemudian hasil akhirnya akan disimpan ke dalam bentuk *file*.

* 1. **Analisis Kebutuhan**

Pada pengembangan aplikasi ini diperlukan beberapa perangkat pendukung sepeti perangkat keras dan perangkat lunak sebagai berikut:

* + 1. **Kebutuhan Perangkat Keras**

Analisis kebutuhan perangkat keras yang digunakan adalah laptop dengan spesifikasi sebagai berikut:

1. Sistem Operasi : Windows 10 64-bit
2. Processor : 11th Gen Intel ® Core ™ i5-1135G7 @2.40 GHz
3. RAM : 16GB
4. Manufaktur : Acer Swift 3 Infinity 4
   * 1. **Kebutuhan Perangkat Lunak**

Aplikasi yang dibangun menggunakan framework Netbeans sebagai teks editor yang digunakan untuk membuat program kriptografi dengan menggunakan bahasa Java.

* + 1. **Kebutuhan Data**

Data yang digunakan adalah data berupa file txt yang kemudian akan diproses untuk diberikan tanda tangan digital.

* + 1. **Prosedur**

Dalam membuat tugas akhir ini, urutan prosedur yang dilakukan dimulai dari studi literatur, analisis kebutuhan, melakukan perancangan program, pembuatan dan pengujian program, dan terakhir dilakukan analisis akhir terhadap aplikasi yang telah berhasil diterapkan kriptografi RSA (Rivest Shamir Adleman) tersebut.

* 1. **Perancangan Aplikasi**

Perancangan program merupakan gambaran proses perencanaan dan pembangunan apikasi. Tujuannya adalah memberikan gambaran cara kerja dari aplikasi ini. Perancangan ini meliputi proses pembentukan message, generate key dari kriptografi RSA (Rivest Shamir Adleman), pembangkitan signature, penggabungan message dan signature, dan dilakukan otentikasi.

1. **Sistem RSA (Rivest Shamir Adleman)** ***Digital Signature***

Aplikasi sistem RSA (Rivest Shamir Adleman) *digital signature* menggunakan algoritma RSA (Rivest Shamir Adleman) pada saat proses *generate key* untuk mendapatkan kunci *public* dan kunci *private*. Kunci ini yang akan digunakan untuk mengenkripsi pesan pada proses *signing*. Berikut adalah contoh algoritma RSA (Rivest Shamir Adleman). Berdasarkan pada bab 2.2 berikut adalah langkah untuk mendapatkan kunci pada saat proses *generate key* dan proses enkripsi yaitu:

1. Memilih bilangan prima yang berbeda, misalkan dengan p dan q dimana p ≠ q maka nilai modulus atau n = p\*q = 7\*13 = 91
2. Hitung nilai phi atau π(n)= (p-1)(q-1) = (7-1)(13-1)= 6\*12= 72.
3. Pilih bilangan e yang akan menjadi kunci publik, diantara 1 sampai 72 dengan syarat gcd atau FPB (e,72) = 1. Coba e = 5 dan dilakukan pengujian apalah gcd (5,72) = 1. Faktor dari 5 = 1 dan 5, sedangkan faktor dari 72 = 1, 2, 3, 4, 6, 8, 9, 12, 18, 24, dan 36. Dapat dilihat bahwa gcd (5,72) = 1. Maka e = 5 bisa digunakan.
4. Kemudian hitung nilai d yang akan menjadi kunci privat, dimana d.e = 1 mod π(n). Maka d.5 = 1 mod 72. Jika 5 dipindahkan ke sebelah kanan persamaan akan didapatkan d = 1/5 mod 72 dan apabila angka 5 di naikkan ke atas akan didapatkan nilai berikut d = 5-1 mod 72. Dengan teorema Euclidean (yang sudah dijelaskan pada bab 2.2) yaitu algoritma untuk mencari PBB dari dua buah bilangan bulat. Maka:

Langkah 1) 72 = 14(5) + 2 p0 = 0

Langkah 2) 5 = 2(2) + 1 p1 = 1

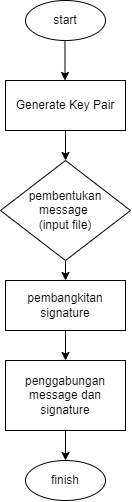
Langkah 3) 2 = 2(1) + 0 p2 = p0 – p1(a1) mod

p2 = 0 – 1(14) mod 72 = 58

p3 = 1 – 58(2) mod 72 = 29

maka d = 29

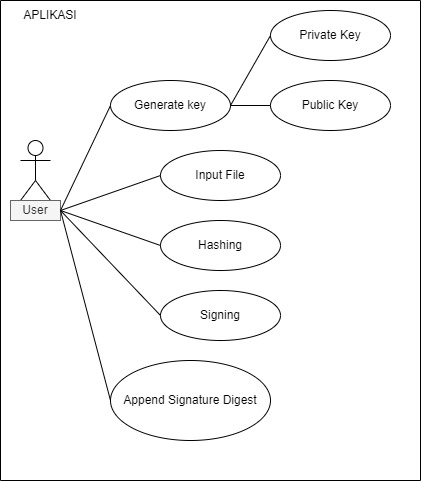
1. Maka didapatkan pasangan kunci publiknya (e,n) = (5,91) dan kunci privatnya (d,n) = (29,91)
2. Untuk mengenkripsi pesan, konversikan pesan ke dalam bilangan bulat m misalkan M = 90. Hitung nilai C = Me mod n dengan e = 5 dan n = 72 yang sudah didapatkan dari proses sebelumnya. Maka C = 905 mod 91 = 90. Nilai ini lah yang menjadi hasil enkripsi.
3. **Flowchat Proses Digital Signature**



Gambar 3. Flowchart proses digital signature

Pada proses ini *generate key pair* untuk menghasilkan sepasang kunci publik dan kunci privat menggunakan algoritma RSA (Rivest Shamir Adleman). Pembuatan *message* dengan menginputkan *file* yang dibutuhkan. Kemudian dilakukan pembangkitan *signature* dengan melakukan *hashing* untuk mendapatkan *message digest* (intisari pesan), setelah itu hasil dari *message digest* digabungkan dengan signature melalui proses enkripsi menggunakan sepasang kunci privat. Hasil ini yang akan menjadi *digital signature*.

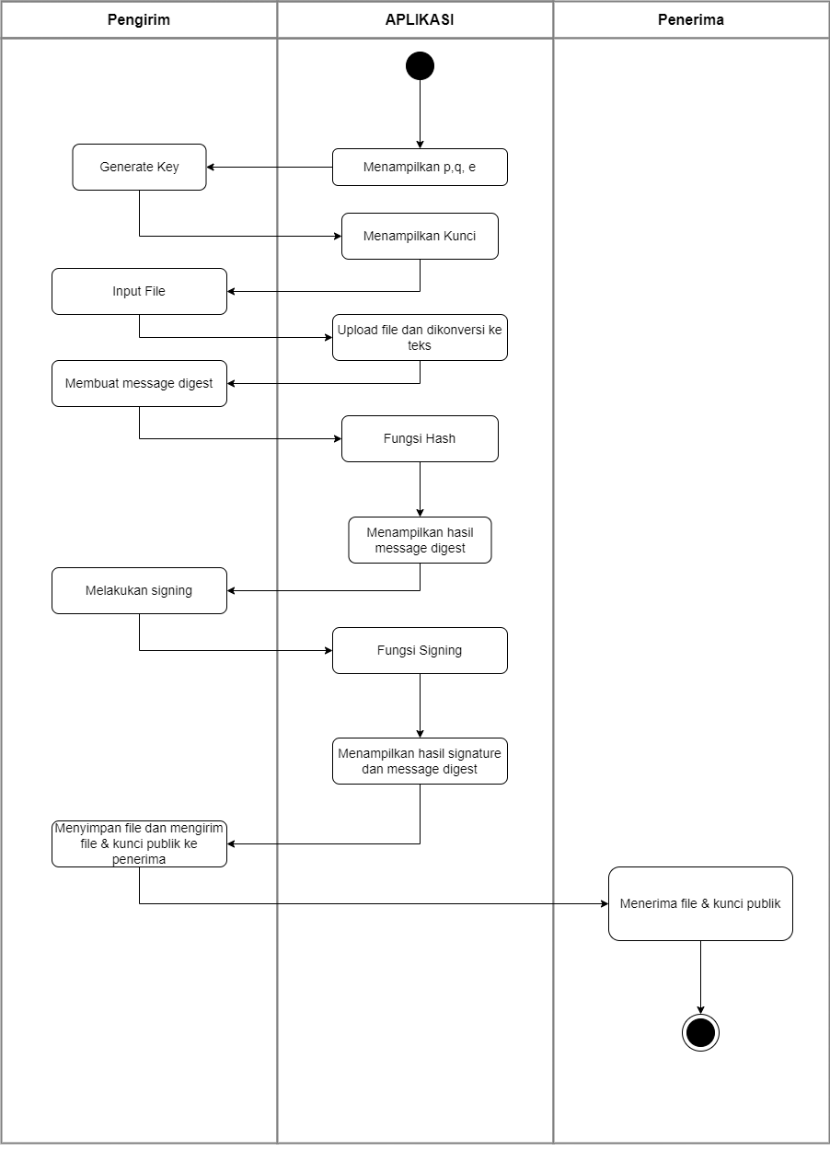
1. **Use *Case diagram***



Gambar 3. *Use Case Diagram*

Use case diagram akan dibuat berdasarkan kondisi perancangan perangkat lunak ini seperti pada Gambar 3.2, terdapat user yang akan generate key untuk mendapatkan kunci publik dan kunci privat, kemudian diinputkan file yang akan diamankan, diberikan signing kemudian dilakukan hashing, terakhir digabungkan dan menghasilkan signature digest.

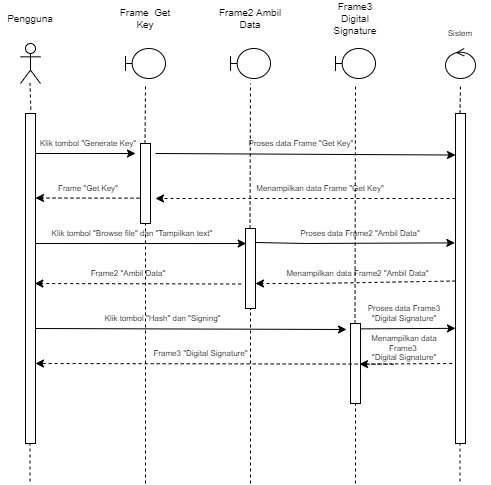
1. **Activity *diagram***



Gambar 3. Avtivity Diagram

Diagram aktivitas menggambarkan aliran kerja dari suatu sistem. Saat pengguna membuka aplikasi, pengguna akan diarahkan untuk melakukan *generate key* untuk mendapatkan kunci. Kemudian pengguna akan diminta memasukkan file yang diinginkan, file tersebut akan dikonversikan ke teks oleh sistem. Lalu pengguna akan melakukan *hashing* untuk mendapatkan *message digest*. Pengguna akan diminta untuk melakukan *signing*. Setelah itu hasil dari *message digest* dan *signature* digabungkan untuk disimpan*.*

1. **Sequence Diagram**

**

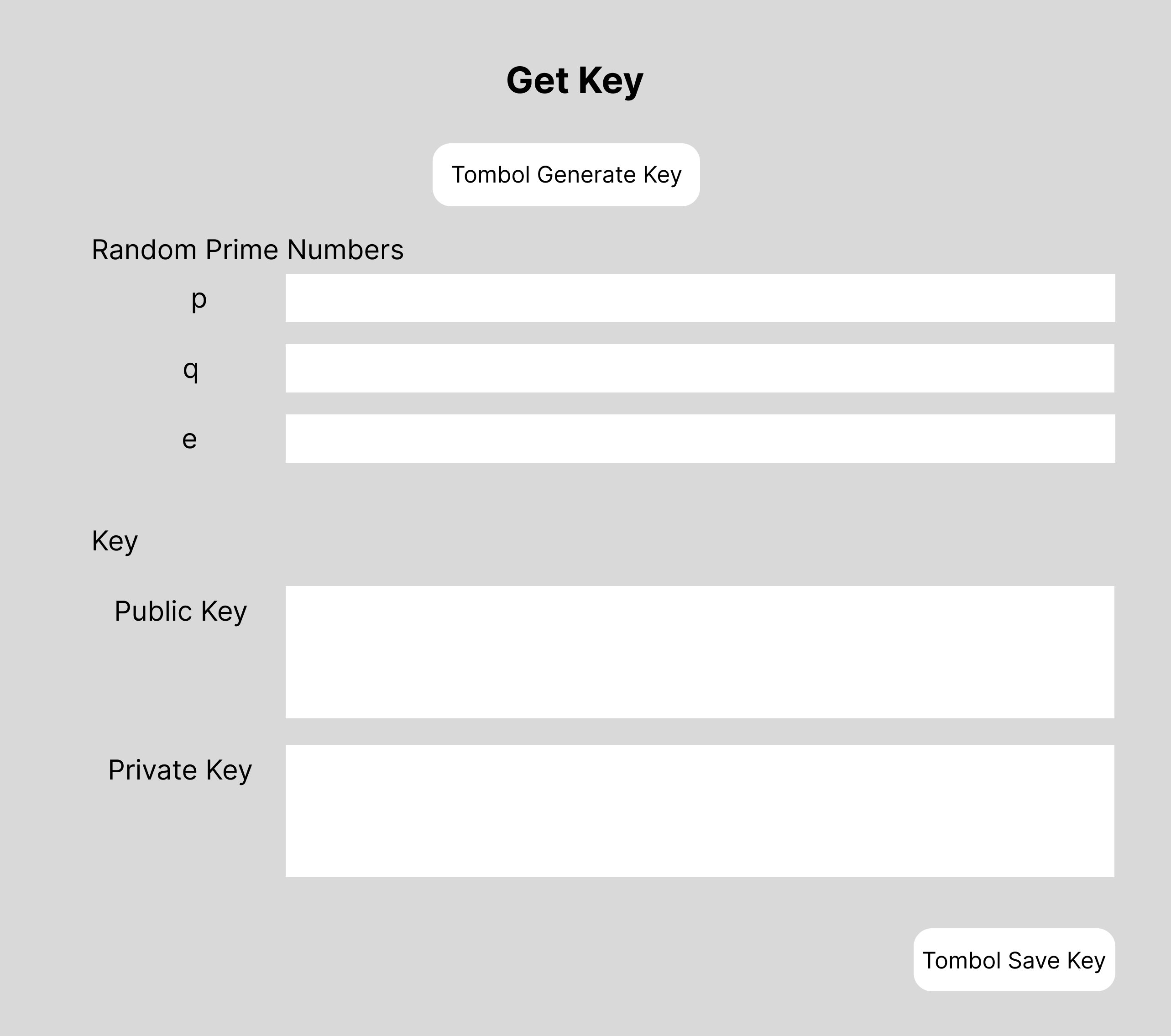
Gambar 3. *Sequence Diagram*

Sequence diagram menunjukkan interaksi yang bisa dilakukan oleh pengguna. Pengguna masuk pada halaman Frame untuk melakukan generate key. Lalu masuk ke Frame2 untuk melakukan memasukkan file dan menampilkan hasil konversinya ke teks. Kemudian masuk ke Frame3 untuk melakukan proses hashing dan append untuk menggabungkan hasil message digest dan signature. Setelah mengakses keseluruhan Frame maka sistem akan memproses data dan menampilkan sebagai halaman yang dipilih oleh pengguna.

* 1. **Desain Tampilan Program**

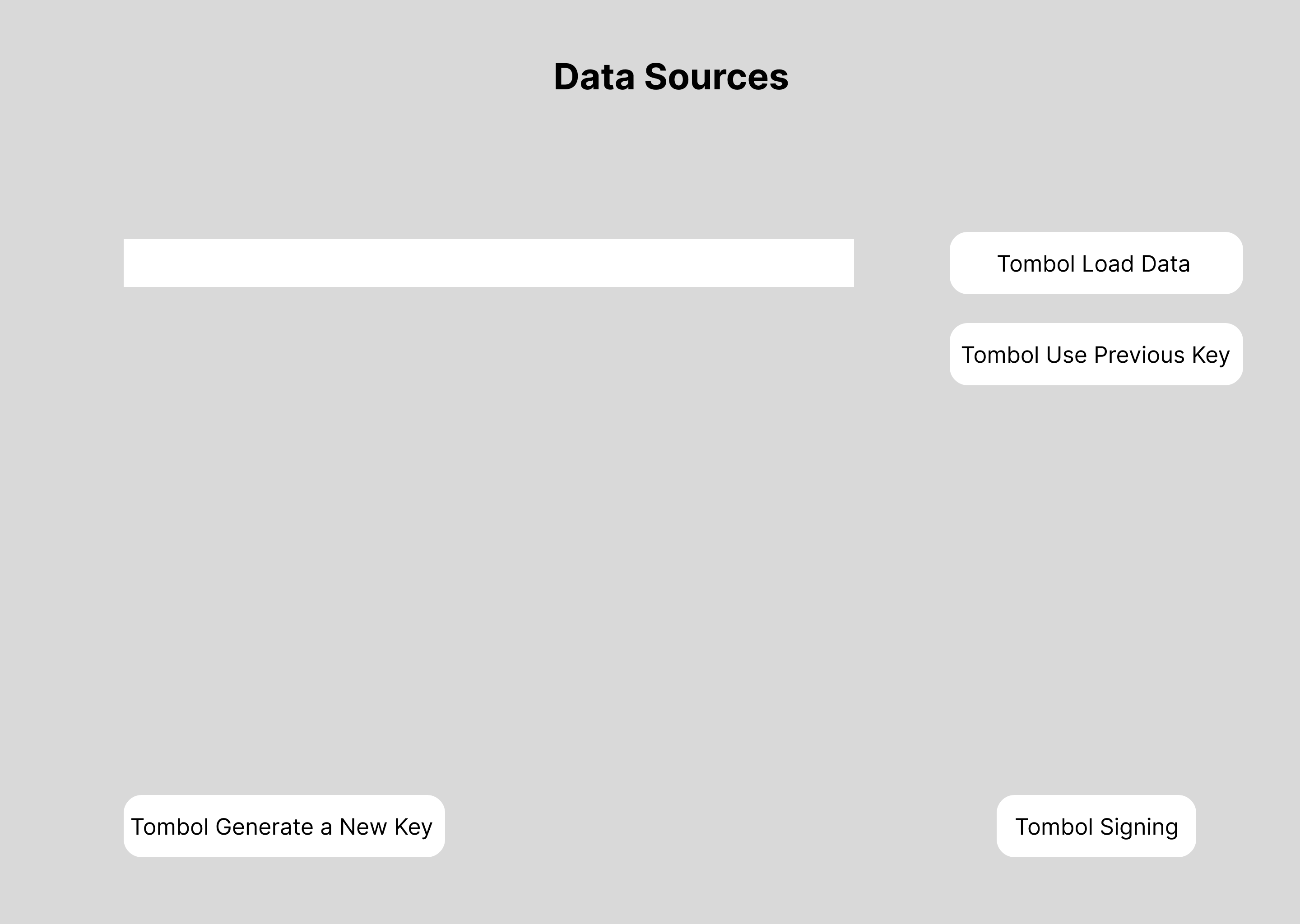
Desain ini dilakukan untuk memudahkan proses implementasi coding dan mengurangi kesalahan dalam proses pengembangan aplikasi. Berikut merupakan desain nya.

Halaman 1, tampilan ini yang akan muncul ketika aplikasi dibuka. Disini menampilkan tombol *generate key* yang bisa diklik oleh pengguna untuk melakukan perhitungan algoritma RSA (Rivest Shamir Adleman) sehingga menampilkan bilangan prima acak p, q, e dan menghasilkan kunci publik dan kunci privat.



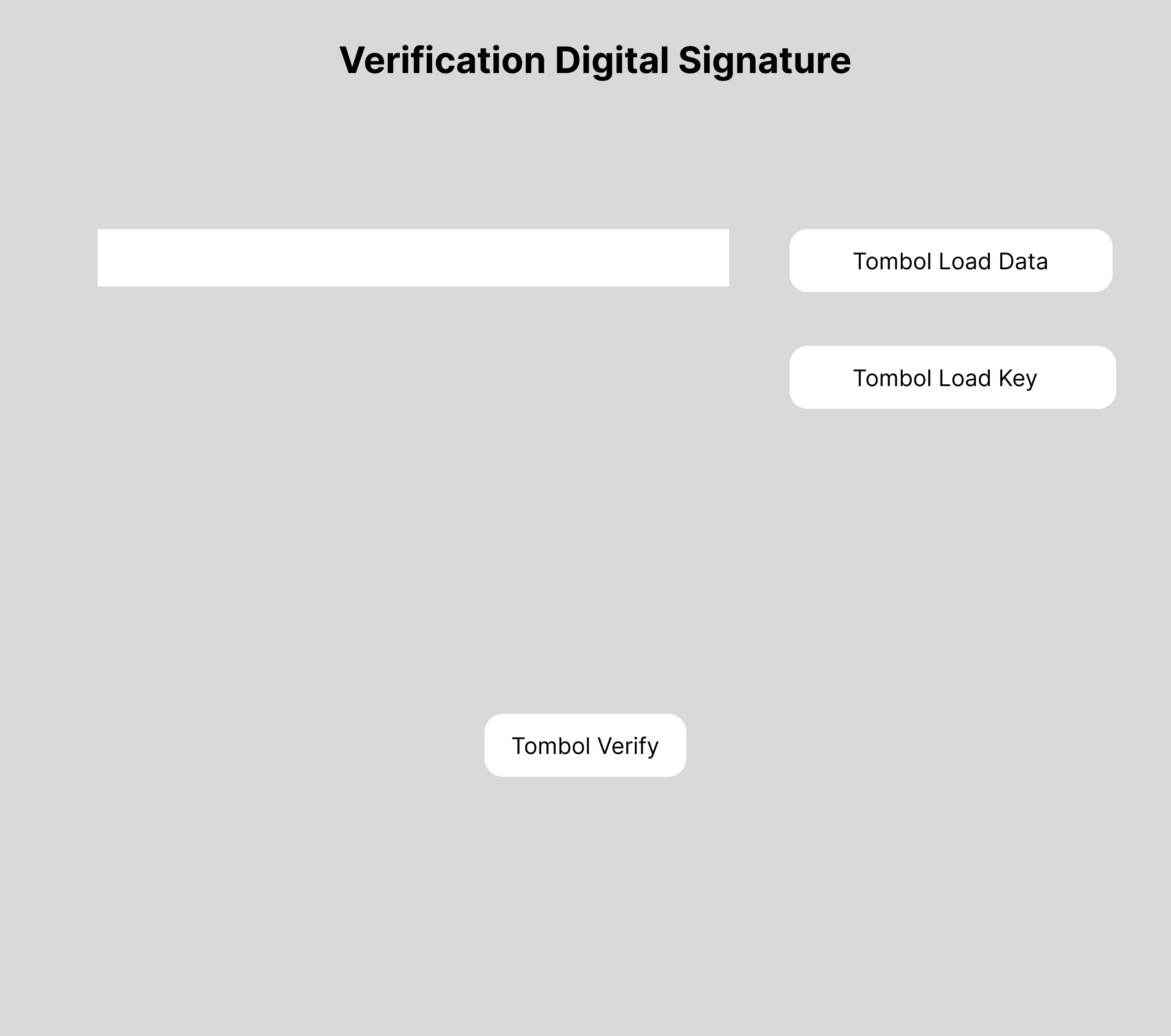
Gambar 3. Halaman 1

Halaman 2 yang merupakan halaman untuk menginputkan file yang akan diamankan. Pada halaman ini juga disediakan tombol *use previous key*, jika pengguna ingin menggunakan kunci yang didapatkan sebelumnya atau bisa menggunakan tombol *generate a new key* jika ingin mengambil kunci baru. Kemudian pengguna bisa menggunakan tombol *signing* untuk memproses *digital signature*.



Gambar 3. Halaman 2

Halaman 3 yang merupakan halaman untuk melakukan verifikasi dari *digital signature*. Pada halaman ini terdapat tombol *Load Data* untuk mengambil file awal yang akan diamankan sebelumnya. Kemudian terdapat tombol *Load Digital Signature* yang digunakan untuk mengambil file hasil *digital signature* yang telah didapatkan dari proses sebelumnya. Terdapat juga tombol *Load Public Key* yang digunakan untuk mengambil kunci untuk digunakan dalam proses verifikasi. Setelah semua komponen terkumpul, dilakukan proses verifikasi dengan mendekripsi *digital signature* menggunakan kunci publik dan hasil nya dibandingkan dengan nilai *hash* yang didapat dari *file* pesan asli.



Gambar 3. Halaman 3

* 1. ***Blackbox Testing***

Pada blackbox testing aplikasi ini, ada beberapa fungsi yang akan diuji, yaitu sebagai berikut :

1. Tombol *Save Key*.
2. Tombol *Load Data*.
3. Tombol *Use Previous Key*.
4. Tombol *Generate a New Key*.
5. Tombol *Signing*.
6. Tombol *Load Data* (2).
7. Tombol *Load Key*.
8. Tombol *Verify*.

# **BAB IV**

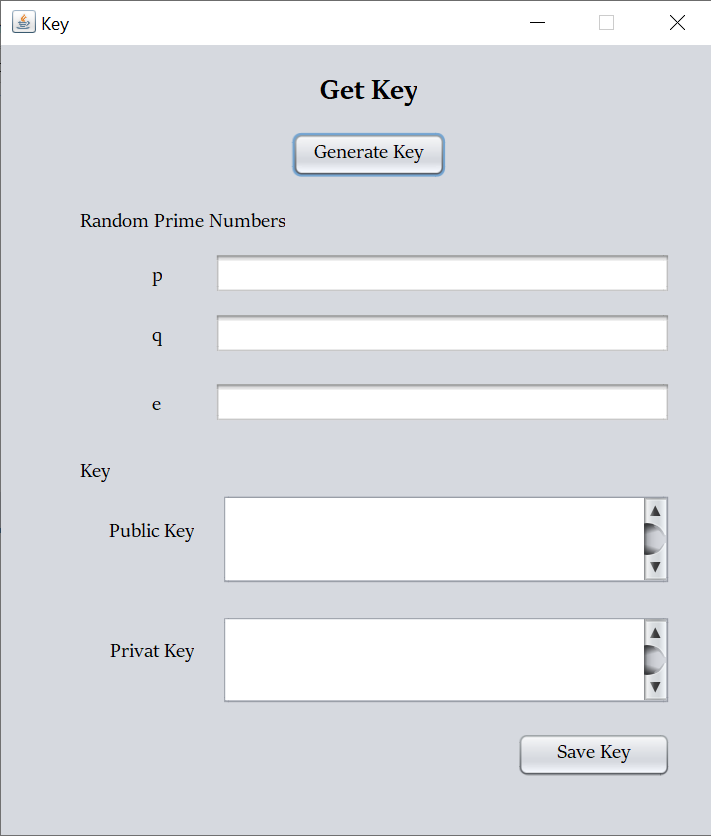
**HASIL DAN PEMBAHASAN**

1. **Impelementasi Program**

Pada tahap ini dilakukan implementasi dari setiap analisis dan rancangan ke dalam kodingan. Penulis menggunakan Java sebagai bahasa pemrograman dan framework netbeans untuk membuat aplikasi ini.

1. **Halaman Pertama**

Saat membuka aplikasi, pengguna langsung diarahkan ke halaman Get Key. Pada halaman ini dilakukan perhitungan algoritma RSA (Rivest Shamir Adleman). Pengguna diminta untuk mengklik tombol generate key untuk mendapatkan kunci publik dan kunci privat.



Gambar 4. Halaman Pertama

Script/kode program halaman pertama saat mengklik tombol *generate key* :

|  |
| --- |
| private void GetKeyActionPerformed(java.awt.event.ActionEvent evt) {  try {  RSA.main(new String[]{});  } catch (NoSuchAlgorithmException ex) {  Logger.getLogger(Halaman1.class.getName()).log(Level.SEVERE, null, ex);  } catch (IOException ex) {  Logger.getLogger(Halaman1.class.getName()).log(Level.SEVERE, null, ex);  }  } |

Tombol *generate key* memanggil fungsi RSA.main dari file RSA.java untuk menjalankan program yang telah dirancang. Berikut adalah kode untuk tombol *save key:*

Script/kode program halaman pertama saat mengklik tombol *save key* :

|  |
| --- |
| private void SaveKeyActionPerformed(java.awt.event.ActionEvent evt) {  if(KPublik.getText().isEmpty() || KPrivat.getText().isEmpty()) {  JOptionPane.showMessageDialog(this, "Klik Generate Key terlebih dahulu!", "Error", JOptionPane.ERROR\_MESSAGE);  return;  }  JFileChooser fileChooser = new JFileChooser();  fileChooser.setDialogTitle("Simpan Kunci");  FileNameExtensionFilter filter = new FileNameExtensionFilter("Text Files", "txt");  fileChooser.setFileFilter(filter);    JOptionPane.showMessageDialog(this, "Masukkan nama file untuk menyimpan kunci", "Info", JOptionPane.INFORMATION\_MESSAGE);  fileChooser.setCurrentDirectory(new File("C:\\Users\\Shania\\OneDrive\\Documents\\NetBeansProjects\\satu\\hasil\\key"));    boolean fileSaved = false;  while (!fileSaved) {  int returnValue = fileChooser.showSaveDialog(this);  if (returnValue == JFileChooser.APPROVE\_OPTION) {  String publicFilePath = fileChooser.getSelectedFile().getAbsolutePath() + "\_public.txt";  String privateFilePath = fileChooser.getSelectedFile().getAbsolutePath() + "\_private.txt";  if (publicFilePath.isEmpty() || privateFilePath.isEmpty()) {  JOptionPane.showMessageDialog(this, "Nama file tidak valid!", "Error", JOptionPane.ERROR\_MESSAGE);  continue;  }  File publicFile = new File(publicFilePath);  File privateFile = new File(privateFilePath);  if (publicFile.exists() || privateFile.exists()) {  int overwriteOption = JOptionPane.showConfirmDialog(this, "Nama file sudah ada. Apakah Anda ingin menimpanya?", "Konfirmasi", JOptionPane.YES\_NO\_OPTION);  if (overwriteOption == JOptionPane.NO\_OPTION) {  continue;  }  }  try (FileWriter publicWriter = new FileWriter(publicFilePath)) {  publicWriter.write("Public key (e,n): " + KPublik.getText());  } catch (IOException ex) {  ex.printStackTrace();  }  try (FileWriter privateWriter = new FileWriter(privateFilePath)) {  privateWriter.write("Private key (d,n): " + KPrivat.getText());  } catch (IOException ex) {  ex.printStackTrace();  }  JOptionPane.showMessageDialog(this, "Kunci telah disimpan", "Info", JOptionPane.INFORMATION\_MESSAGE);  fileSaved = true;    Hal2 hal2 = new Hal2();  hal2.setVisible(true);  this.setVisible(false);  }else {  JOptionPane.showMessageDialog(this, "Simpan kunci dibatalkan", "Info", JOptionPane.INFORMATION\_MESSAGE);  fileSaved = true;  }  } |

Pada kode diatas dirancang fungsi untuk menyimpan value dari kunci publik dan kunci privat yang telah didapatkan dari proses perhitungan algoritma RSA (Rivest Shamir Adleman) untuk kemudian disimpan ke dalam file txt dengan nama file yang bisa di inputkan oleh *user*.

1. **Halaman** **Kedua**

Setelah klik tombol *save key* pada halaman pertama, akan langsung diarahkan ke halaman kedua. Pada halaman ini pengguna akan memilih file yang akan dimasukkan kemudian diminta untuk mengklik tombol *use a previous key* untuk memanggil value kunci yang didapatkan sebelumnya. Disediakan juga pilihan untuk mendapat kunci baru jika diinginkan pada tombol *generate a new key*.



Gambar 4. Halaman Kedua

Script/kode program halaman kedua saat mengklik tombol *Load Data* :

|  |
| --- |
| private void LoadDataActionPerformed(java.awt.event.ActionEvent evt) {  JFileChooser dialog = new JFileChooser();  dialog.setCurrentDirectory(new File("C:\\Users\\Shania\\OneDrive\\Documents\\NetBeansProjects\\satu\\hasil\\sources"));  int result = dialog.showOpenDialog(this);    if (result == JFileChooser.APPROVE\_OPTION) {  File file = dialog.getSelectedFile();  try {  String filePath = file.getAbsolutePath();  txtPATH.setText(filePath);  } catch (Exception e) {  JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memilih file:\n" + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);  e.printStackTrace();  }  }  } |

*CurrentDirectory* digunakan untuk memanggil folder terakhir yang berada bersama folder kodingan aplikasi ini. Lalu dipilih file yang diinginkan untuk ditampilkan pada txtPath.setText. Berikut adalah kode untuk tombol *display contents:*

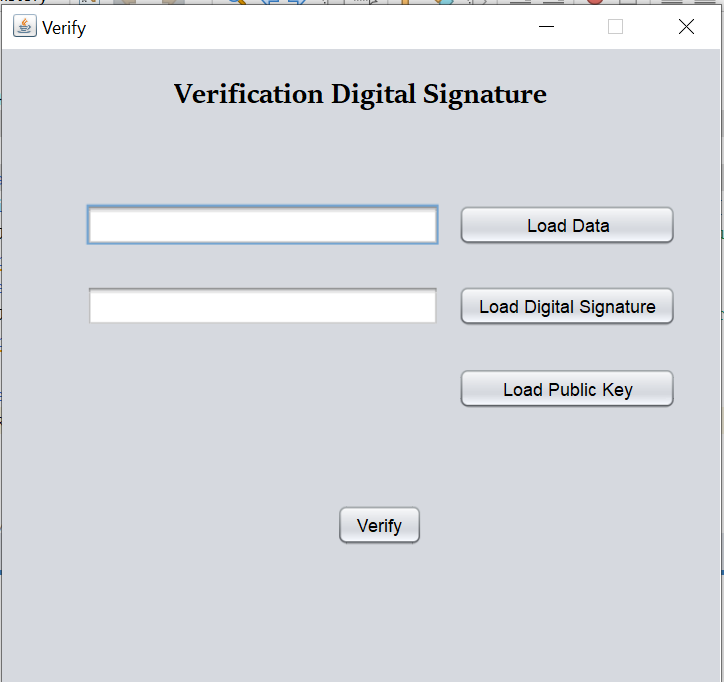
Script/kode program saat mengklik tombol *use previous key* :

|  |
| --- |
| private void PrevKeyActionPerformed(java.awt.event.ActionEvent evt) {  if(PrevKey.isSelected()){  String privateKey = Halaman1.KPrivat.getText();  } else{  JOptionPane.showMessageDialog(this, "Tombol belum diaktifkan!", "Error", JOptionPane.ERROR\_MESSAGE);  }  } |

Untuk memanggil *value* dari kunci yang sudah didapatkan sebelumnya.

1. **Halaman Ketiga**

Setelah klik tombol next pada halaman kedua, akan langsung diarahkan ke halaman ketiga. Pada halaman ini, pengguna diminta untuk mengklik tombol hash untuk mendapatkan message digest, lalu hasilnya akan ditampilkan bersama dengan kunci privat yang telah didapatkan dari proses perhitungaan algoritma RSA (Rivest Shamir Adleman) pada halaman pertama. Lalu tombol signing akan mengenkripsi message digest menggunakan kunci privat untuk menghasilkan digital signature. Hasil akhirnya akan disimpan ke dalam file txt.



Gambar 4. Halaman Ketiga

Script/kode program halaman ketiga saat mengklik tombol load data :

|  |
| --- |
| private void LDataActionPerformed(java.awt.event.ActionEvent evt) {  dialog.setCurrentDirectory(new File("C:\\Users\\Shania\\OneDrive\\Documents\\NetBeansProjects\\satu\\hasil\\sources"));  int pfile = dialog.showOpenDialog(this);    if(pfile == JFileChooser.APPROVE\_OPTION){  File file = dialog.getSelectedFile();  try {  tdata.setText("" +file);  } catch (Exception e){  System.out.println("Error :" +e.getMessage());  }  }  } |

txtContent.setText digunakan untuk memanggil value content yang telah didapatkan pada proses yang dilakukan pada halaman kedua. bilKey.setText digunakan untuk memanggil value kunci publik yang telah didapatkan pada proses yang dilakukan pada halaman pertama. Kemudian dilakukan fungsi RSA.main untuk memproses kodingan yang telah dirancang pada RSA.java. Berikut adalah kode untuk tombol signing:

Script/kode program saat mengklik tombol signing :

|  |
| --- |
| private void SignActionPerformed(java.awt.event.ActionEvent evt) {  Halaman4 hal4 = new Halaman4();  Halaman4.txtFinal.getText();  hal4.setVisible(true);  this.dispose();  try {  RSA.main(new String[]{});  } catch (NoSuchAlgorithmException ex) {  Logger.getLogger(Halaman1.class.getName()).log(Level.SEVERE, null, ex);  }  } |

Pada kodingan diatas, diproses fungsi RSA.main yang telah dirancang pada RSA.java untuk menghasilkan proses enkripsi sehingga menghasilkan digital signature.

1. **Pengujian**

Pengujian yang dilakukan pada aplikasi ini adalah blackbox testing yang merupakan jenis testing yang bertujuan untuk melihat fungsional dari program yang dikembangkan. Blackbox testing ini hanya memperhatikan masukkan ke dalam sistem dan keluaran dari masukkan tersebut. Berikut hasil pengujiannya:

|  |  |  |  |
| --- | --- | --- | --- |
| **Fungsi Uji** | **Skenario** | **Hasil yang Diharapkan** | **Hasil Pengujian** |
| Tombol *Generate Key* | Menekan tombol *generate key* | Sistem akan menghitung nilai p dan q sehingga menampilkan kunci publik dan kunci privat | Berhasil |
| Tombol *Browse File* | Menekan tombol *browse file* dan memilih file yang diinginkan | Dokumen file yang dipilih akan ditampilkan | Berhasil |
| Tombol Tampilkan *Text* | Menekan tombol tampilkan *text* | Menampilkan hasil dokumen yang telah dikonversi ke teks | Berhasil |
| Tombol *Hash* | Menekan tombol *hash* | Dokumen yang telah dikonversi akan dihitung nilai *hash*nya untuk mendapatkan *message digest* |  |
| Tombol *Signing* | Menekan tombol *signing* | Menggabungkan *signature* dengan *message digest* melalui proses enkripsi |  |

Tabel 4. Blackbox Testing

# **BAB V**

**KESIMPULAN DAN SARAN**

1. **Kesimpulan**

Berdasarkan berbagai hal yang telah disampaikan pada bab sebelumnya serta perancangan dan implementasi web repositori yang sudah dilakukan penulis, maka dapat diambil simpulan sebagai berikut:

Tujuan dari pembuatan aplikasi ini tercapai karena kriptografi RSA (Rivest Shamir Adleman) berhasil diterapkan dan bekerja baik dengan proses enkripsinya.

Implementasi dari digital signature berhasil dilakukan sehingga data dapat dijaga keamanannya.

……….. ………….. . ……….. ……….. ……… ………… …………. ……. ……….. …….. . ……….. …….. ……….. .

1. **Saran**

Berdasarkan penelitian yang dilakukan, penulis memberikan saran yang dapat

diimplementasikan pada penelitian selanjutnya, yaitu sebagai berikut:

1. 1. ……….. ………….. . ……….. ……….. ……… ………… …………. ……. ……….. …….. . ……….. …….. ……….. .
2. 2. ……….. ………….. . ……….. ……….. ……… ………… …………. ……. ……….. …….. . ……….. …….. ……….. .
3. 3. ……….. ………….. . ……….. ……….. ……… ………… …………. ……. ……….. …….. . ……….. …….. ……….. .

# **DAFTAR PUSTAKA**

Munir, R. (2019). Kriptografi. Bandung: Informatika..

Pial, H. R. (2020). *Digital Signature Understanding. How it Works and Importance*. https://doi.org/10.13140/RG.2.2.20694.06720.

Denning, D. E. (1992). *Cryptography and data security*. Mass.: Addison-Wesley.

Garfinkel, S. and Lipford, H. R. (2014) Usable Security: History, Themes, and Challenges. Morgan & Claypool Publishers.

Gunawan, I. (2020). Keamanan data: Teori dan implementasi. Available at: https://books.google.co.id/books?id=5RjKDwAAQBAJ (Accessed: 12 March 2022).

Middleton, B. (2017) *A History of Cyber Security Attack: 1980 to Present.* CRC Press.

Azdy, R. A. (2016). Tanda tangan Digital Menggunakan Algoritme Keccak dan RSA. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi (JNTETI)*, *5*(3).

H. Kridalaksana, A., Rangan, A. Y., & Ansharie, A. (2017). ENKRIPSI DATA AUDIO MENGGUNAKAN METODE KRIPTOGRAFI RSA. *Sebatik*, *17*(1), 6–10.

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. https://doi.org/10.1109/tit.1976.1055638

Pieprzyk, J., & Sadeghiyan, B. (1993). *Design of Hashing Algorithms*. Springer Verlag.

Preneel, B. (2010). Cryptographic hash functions. *European Transactions on Telecommunications*, *5*(4), 431–448. https://doi.org/10.1002/ett.4460050406

Stinson, D. R., & Paterson, M. B. (2019). *Cryptography : theory and practice*. Crc Press.

Gupta, A., & Anand, A. (2017). Ethical Hacking and Hacking Attacks. *International Journal of Engineering and Computer Science*, *6*(4). https://doi.org/10.18535/ijecs/v6i4.42

# **LAMPIRAN**

**Lampiran 1 Ringkasan Hasil *Usability Testing***

**Lampiran 2** **Kode Web Aplikasi**

Register

|  |
| --- |
| import React, { Component } from 'react'  import bg2 from '../icons/bg2.webp'  import axios from 'axios'  import { scrollToTop } from '../helpers/autoScroll'  import { ProgressBar, Modal} from 'react-bootstrap'  import { Redirect } from 'react-router-dom'  import { connect } from 'react-redux'  export class Register extends Component {  initialState = {  showLoading:false,  name:'',  npm: '',  email:'',  pass:'',  rePass: '',  message: '',  status:'',  displayForm1:'block',  displayForm2:'none',  displayForm3:'none',  file:null,  progress:34  }  state=this.initialState  next = (e) =>{  e.preventDefault()  this.setState({  showLoading:true,  })  let {name, npm, pass, email}= this.state  let data={  name:name,  email:email,  npm:npm,  password:pass  }  axios({  method: 'POST',  url: '/check-form',  data: data  }).then(res => {  this.setState({  message: '',  displayForm1: 'none',  displayForm2: 'block',  progress: this.state.progress + 33,  showLoading:false  })  scrollToTop()  }).catch(err => {  this.setState({  showLoading:false  })  if (err.response) {  this.setState({  message: err.response.data.message,  status: err.response.data.status,  })  }  else{  this.setState({  message: 'Network error, Cek Koneksi Anda',  status: 500,  })  }  })  } |

*Login*

|  |
| --- |
| import React, { PureComponent } from 'react'  import UserMenu from '../components/UserMenu'  import AdminMenu from '../components/AdminMenu'  import { Link, Redirect } from 'react-router-dom'  import { setToken, delToken} from '../reducers/authReducer'  import { connect } from 'react-redux'  import { Modal } from 'react-bootstrap'  import { FaRegCheckCircle } from 'react-icons/fa'  import '../styles/nav.css'  import axios from 'axios'  import { scrollToTop } from '../helpers/autoScroll'  import MediaQuery from 'react-responsive'  export class Nav extends PureComponent {  state = {  npm: '',  pass: '',  message: '',  status: null,  showLogin: false,  showLoading:false,  justLoggedIn:false, //ketika pertama kali login, agar refresh tidak redirect  }  handleInput = (e) =>{  this.setState({  status:'',  [e.target.id] : e.target.value  })  }  submitLogin = e => {  e.preventDefault()  this.setState({  showLoading:true  })  axios({  method: 'post',  url: '/login',  data: {  npm: this.state.npm,  password: this.state.pass  }  }).then(res => {  let loginInfo = res.data.data  console.log(loginInfo)  if (loginInfo.isLogged){ //response didapat  this.props.login(loginInfo) //set state global  this.setState({ //show modal  showLoading:false,  status:res.data.status,  justLoggedIn:true,  showLogin:true,  })  scrollToTop()  setTimeout(() =>  this.setState({ //hide modal  showLogin:false  }), 1000)  }  else{  this.setState({  showLoading:false,  status:500  })  }  }).catch((err) => {  if(err.response){  this.setState({  message:err.response.data.message,  status:err.response.data.status,  showLoading:false,  })  } else{  this.setState({  status: 500,  showLoading:false,  })  }  })  }  logout = () =>{  this.props.logout()  }  handleClose = () => {  this.setState({  showLogin:false  })  } |

Unggah

|  |
| --- |
| import React, { Component } from 'react'  import { connect } from 'react-redux'  import { Redirect, Link } from 'react-router-dom'  import axios from 'axios'  import { Spinner, Modal } from 'react-bootstrap'  export class Upload extends Component {  initialState={  skripsi:{},  isLoaded:false,  offline: false,  showLoading:false,  title:'',  titleAlert:'initial',  year:'',  yearAlert:'initial',  abstract:'',  abstractAlert:'initial',  abstrak:'',  abstrakAlert:'initial',  category:'',  keywords:'',  file:null,  message: '',  status:'',  }  state=this.initialState  submit = (e) =>{  e.preventDefault()  this.setState({  showLoading:true  })  let {title, year, abstrak, abstract, category, keywords} = this.state  let {file} = this.state  const formData = new FormData()  formData.append('file', file)  formData.append('title', title)  formData.append('year', year)  formData.append('abstrak', abstrak)  formData.append('abstract', abstract)  formData.append('category', category)  formData.append('keywords', keywords)  axios({  method: 'POST',  url: `/user/upload/`,  data: formData,  headers:{  'Content-Type':'multipart/form-data',  'Authorization': this.props.token  }  }).then((res) =>{  this.refs.uploadForm.reset()  this.setState({  message:res.data.message,  status:res.data.status,  showLoading:false  })  }).catch((err) => {  this.setState({  showLoading:false  })  if( err.response){  this.setState({  message:err.response.data.message,  status:err.response.data.status,  })  }  })  }  handleInput = (e) =>{  if(e.target.id==='title'){  e.target.value=e.target.value.replace(/\n/g, ' ')  }  this.setState({  [e.target.id] : e.target.value,  })  if (e.target.id==='title'){  this.setState({  titleAlert: e.target.value,  })  }  else if (e.target.id==='year'){  this.setState({  yearAlert: e.target.value,  })  }  else if (e.target.id==='abstract'){  this.setState({  abstractAlert: e.target.value,  })  }  else if (e.target.id==='abstrak'){  this.setState({  abstrakAlert: e.target.value,  })  }  } |

Detail Skripsi

|  |
| --- |
| import React, { Component } from 'react'  import { Spinner } from 'react-bootstrap'  import axios from 'axios'  import { connect } from 'react-redux'  import { FaChevronLeft, FaChevronRight } from 'react-icons/fa'  import {Document, pdfjs, Page} from 'react-pdf'  pdfjs.GlobalWorkerOptions.workerSrc = `//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjs.version}/pdf.worker.js`  export class SkripsiDetail extends Component {  state={  skripsi:[],  isLoaded:false,  offline:false,  pageNumber:1,  numPages:null  }  getData =()=>{  let id = this.props.match.params.id  if(this.props.token){  axios({  method: 'get',  url: `/skripsi/detail/`,  params:{  id : id  },  headers: {  Authorization: this.props.token  }  }).then(res=>{  this.setState({  skripsi: res.data[0],  isLoaded: true  })  }).catch(err=>{  if(err.response){  console.log(err.response)  }  })  }  else{  axios({  method: 'get',  url: `/skripsi/info/`,  params:{  id : id  }  }).then(res=>{  this.setState({  skripsi: res.data[0],  isLoaded: true  })  }).catch(err=>{  if(err.response){  console.log(err.response)  }  })  }  }  onDocumentLoadSuccess = ({ numPages }) => {  this.setState({ numPages });  }  next = () => {  let {pageNumber, numPages} = this.state  if( pageNumber<=numPages){  this.setState({pageNumber:this.state.pageNumber+1})  }  }  before = () => {  let {pageNumber} = this.state  if( pageNumber>1){  this.setState({pageNumber:this.state.pageNumber-1})  }  }  componentDidMount(){  if (navigator.onLine){  this.getData()  this.setState({  offline:false  })  }  else{  this.setState({  offline:true,  })  }  } |

Profil dan Edit *Password*

|  |
| --- |
| import React, { Component } from 'react'  import { Spinner, Modal} from 'react-bootstrap'  import { connect } from 'react-redux'  import axios from 'axios'  import moment from 'moment'  export class ProfileInfo extends Component {  state={  user:{},  isLoaded:false,  showLoading:false,  offline:false,  newPass:'',  oldPass:'',  confirmPass:'',  message:'',  status:null  }  getProfile= ()=>{  axios({  method: 'get',  url: `/user/profile/`,  headers: {  Authorization: this.props.token  }  }).then(res=>{  this.setState({  user: res.data,  isLoaded: true  })  }).catch(err=>{  console.log(err.response)  })  }  componentDidMount(){  if (navigator.onLine){  this.getProfile()  this.setState({  offline:false  })  }  else{  this.setState({  offline:true,  })  }  } |

Status Skripsi

|  |
| --- |
| import React, { Component } from 'react'  import { connect } from 'react-redux'  import axios from 'axios'  import {Spinner} from 'react-bootstrap'  import { FaCheck, FaFilePdf, FaTimes } from 'react-icons/fa'  import { Link } from 'react-router-dom'  import moment from 'moment'  export class SkripsiStatus extends Component {  state={  skripsi:{},  isLoaded:false,  Offline:false  }  getSkripsi=()=>{  axios({  method: 'get',  url: `/user/skripsi/`,  headers: {  Authorization: this.props.token  }  }).then(res=>{  this.setState({  skripsi: res.data,  isLoaded: true  })  }).catch(err=>{  if(err.response){  console.log(err.response)  }  })  }  componentDidMount(){  if (navigator.onLine){  this.getSkripsi()  this.setState({  offline:false  })  }  else{  this.setState({  offline:true,  })  }  } |

Kontak Admin

|  |
| --- |
| import React, { Component } from 'react'  import { Redirect } from 'react-router-dom'  import { Spinner } from 'react-bootstrap'  import { connect } from 'react-redux'  import axios from 'axios'  import Forum from '../components/Forum'  class UserForum extends Component {  state={  text:'',  dataLoaded:false,  isLoading:false,  chats:[],  offline:false,  message:''  }  handleText = e =>{  this.setState({  [e.target.id] : e.target.value.trim()  })  }  submit = e => {  let {text} =this.state  e.preventDefault()  if(text){  this.setState({  isLoading:true  })  axios({  method: 'post',  url: '/user/insert-text',  headers: {  Authorization:this.props.token  },  data: {  text: text,  }  }).then(res => {  this.setState({  isLoading:false,  text:''  })  this.getForum()  this.refs.messages.reset()  }).catch(err=>{  console.log(err.response)  if(err.response){  this.setState({  message:err.response.data.message  })  }  this.setState({  isLoading:false,  })  setTimeout(() =>  this.setState({  message:''  }), 5000)  })  }  }  getForum= ()=>{  axios({  method: 'get',  url: '/user/forum',  headers: {  Authorization:this.props.token  }  }).then(res=>{  this.setState({  dataLoaded:true,  chats:res.data  })  }).catch((err) => {  if(err.response){  console.log(err.response.statusText)  }  this.setState({  dataLoaded:true  })  })  }  componentDidMount(){  if (navigator.onLine){  this.getForum()  this.setState({  offline:false  })  }  else{  this.setState({  offline:true,  })  }  } |

Lupa *Password*

|  |
| --- |
| import React, { Component } from 'react'  import { Spinner } from 'react-bootstrap'  import { connect } from 'react-redux'  import { Redirect } from 'react-router-dom'  import Bg3 from '../components/Bg3'  import axios from 'axios'  export class Forgot extends Component {  state={  email:'',  npm:'',  checknpm:'',  displaySection1:'block',  displaySection2:'none',  sending:false,  status:'',  message:''  }  handleInput = e => {  this.setState({  [e.target.id] : e.target.value,  })  }  checkNpm = e =>{  this.setState({  checknpm : e.target.value,  })  }  sendEmail = e =>{  e.preventDefault()  this.setState({  sending:true  })  axios({  method: 'put',  url: '/forgot-pass',  data: {  npm: this.state.npm,  email: this.state.email  }  }).then(res => {  this.refs.forgotForm.reset()  this.setState({  sending:false,  status:res.data.status,  message:res.data.message,  displaySection1:'none',  displaySection2:'block'  })  }).catch((err) => {  console.log(err.response)  if(err.response){  this.setState({  status:err.response.data.status,  message:err.response.data.message,  sending:false,  })  }  else{  this.setState({  message: 'Network error, Cek Koneksi Anda',  status: 500,  sending:false,  })  }  })  } |

PWA

|  |
| --- |
| //Lifecycle = register -> install -> activate https://bit.ly/CRA-PWA  //ip localhost ([::1] - 127.0.0.0/8 )  const isLocalhost = Boolean(  window.location.hostname === 'localhost' ||  window.location.hostname === '[::1]' ||  window.location.hostname.match(  /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/  )  )  export function register(config) {  //Cek apakah service worker support pada browser  if ('serviceWorker' in navigator) {  const publicUrl = new URL(process.env.PUBLIC\_URL, window.location.href)  //Service worker tidak jalan jika public file diluar url  if (publicUrl.origin !== window.location.origin) {  return  }  //Ketika halaman telah di load  window.addEventListener('load', () => {  const swUrl = `${process.env.PUBLIC\_URL}/sw-custom.js`  if (isLocalhost) {  checkValidServiceWorker(swUrl, config)  navigator.serviceWorker.ready.then(() => {  console.log('Web app served cache-first')  })  } else {  //Register SW  registerValidSW(swUrl, config)  }  })  }  }  function registerValidSW(swUrl, config) {  //------Register SW  navigator.serviceWorker  .register(swUrl)  .then(registration => {  registration.onupdatefound = () => {  const installingWorker = registration.installing  if (installingWorker == null) {  return  }  installingWorker.onstatechange = () => {  //------------install SW  if (installingWorker.state === 'installed') {  //Cek apakah sw aktif  if (navigator.serviceWorker.controller) {  // Update precached content di fetch tapi SW lama masih di jalankan sampai tab di tutup  console.log('Content baru telah di fetch dan siap digunakan saat tab telah ditutup')  if (config && config.onUpdate) {  config.onUpdate(registration)  }  } else {  console.log('Content telah di cached untuk mode offline.')  if (config && config.onSuccess) {  config.onSuccess(registration)  }  }  }  }  }  })  .catch(error => {  console.error('Error saat register service worker:', error)  })  }  function checkValidServiceWorker(swUrl, config) {  // Cek apakah SW sudah ada  fetch(swUrl, {  headers: { 'Service-Worker': 'script' }  })  .then(response => {  const contentType = response.headers.get('content-type')  if (response.status === 404 || (contentType != null && contentType.indexOf('javascript') === -1)) {  // SW tidak di temukan  navigator.serviceWorker.ready.then(registration => {  registration.unregister().then(() => {  window.location.reload()  })  })  } else {  // SW ditemukan. Proses dilanjutkan  registerValidSW(swUrl, config)  }  })  .catch(() => {  console.log('Tidak ada koneksi Internet. Web App dalam offline mode.')  })  }  export function unregister() {  if ('serviceWorker' in navigator) {  navigator.serviceWorker.ready.then(registration => {  registration.unregister()  })  }  } |

# **RIWAYAT HIDUP**

**DATA PRIBADI**

Nama Lengkap : Shania Salsabila

NPM : 140810180014

Tempat, Tanggal Lahir : Payakumbuh, 24 Januari 2000

Jenis Kelamin : Perempuan

Agama : Islam

Alamat : Jl. H. Rasul no 12 Lingkungan Tarok,Payakumbuh

Telepon : 081268541964

Email : [shaniaslsbl@gmail.com](mailto:shaniaslsbl@gmail.com)

**RIWAYAT PENDIDIKAN**

|  |  |
| --- | --- |
| **Tahun** | **Instansi** |
| 2005 - 2006 | TK S PIUS |
| 2006 - 2012 | SD S PIUS |
| 2012 - 2015 | SMP S FIDELIS |
| 2015 - 2018 | SMA N 2 Payakumbuh |
| 2018 - Sekarang | Universitas Padjadjaran |

**RIWAYAT KEPANITIAAN**

|  |  |  |
| --- | --- | --- |
| **Nama Kepanitiaan** | **Jabatan** | **Tahun** |
| Peluit Himatif | Staff Divisi Dana Konsumsi | 2018 |
| Instagram Himatif | Staff Divisi Medik | 2018 |
| CBS Himatif | Ketua Divisi Dana dan Konsumsi | 2019 |
| Informatics Festival Himatif | Staff Divisi Logistik | 2020 |
| CBS Himatif | Staff Divisi Pembimbing Kelompok | 2020 |
| Padjadjaran Education Festival | Staff Divisi Keamanan | 2021 |