

**PENGHITUNGAN DAN VALIDASI SUARA DALAM SISTEM
E-VOTING DENGAN *HOMOMORPHIC ENCRYPTION* DAN
PROTOKOL *ZERO-KNOWLEDGE PROOF***

SKRIPSI

ATIKHA AZMILA

201401039



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2023**

**PENGHITUNGAN DAN VALIDASI SUARA DALAM SISTEM
E-VOTING DENGAN *HOMOMORPHIC ENCRYPTION* DAN
PROTOKOL *ZERO-KNOWLEDGE PROOF***

SKRIPSI

**Diajukan untuk melengkapi tugas dan memenuhi syarat
memperoleh ijazah
Sarjana Ilmu Komputer**

ATIKHA AZMILA

201401039



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2023**

PERSETUJUAN

Judul : PENGHITUNGAN DAN VALIDASI SUARA DALAM
SISTEM E-VOTING DENGAN HOMOMORPHIC
ENCRYPTION DAN PROTOKOL
ZERO-KNOWLEDGE PROOF

Kategori : SKRIPSI

Nama : ATIKHA AZMILA

Nomor Induk Mahasiswa : 201401039

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Telah diuji dan dinyatakan lulus di Medan, 10 Januari 2024

Pembimbing II



Herryance S.T., M.Kom.
NIP. 198010242010121002


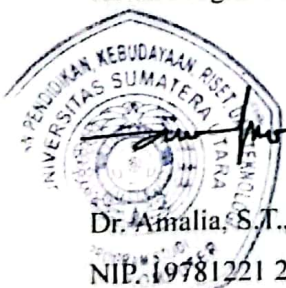
Pembimbing I



Dr. M.Andri Budiman S.T., M.Comp.Sc.,
M.E.M.
NIP. 197510082008011011

Diketahui/Disetujui Oleh

Ketua Program Studi S-1 Ilmu Komputer

Dr. Amalia S.T., M.T
NIP. 19781221 201404 2 001

UNIVERSITAS SUMATERA UTARA

PERNYATAAN**PENGHITUNGAN DAN VALIDASI SUARA DALAM SISTEM E-VOTING
DENGAN HOMOMORPHIC ENCRYPTION DAN PROTOKOL ZERO-
KNOWLEDGE PROOF
SKRIPSI**

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, Desember 2023



Atikha Azmila

201401039

PENGHARGAAN

Segala puji bagi Allah Azza Wa Jalla yang telah melimpahkan keberkahan, rahmat, taufik, inayah, dan hidayahNya kepada penulis sehingga mampu menyelesaikan penyusunan skripsi ini sebagai syarat untuk memperoleh gelar Sarjana Komputer, pada Program Studi S1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara. Shalawat berangkaikan salam kepada baginda Rasulullah, Nabi Muhammad Shallallahu ‘alaihi Wasallam, karena perantara beliau-lah kita bisa merasakan nikmat ilmu pengetahuan yang diridho Allah.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada:

1. Bapak Dr. Muryanto Amin, S.Sos, M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia ST., M.T. selaku Ketua Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Pak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Wakil Dekan 1 Fakultas Ilmu Komputer dan Teknologi Informasi dan dosen pembimbing I yang telah memberikan bimbingan, kritik, motivasi, dan saran serta arahan menuju kebaikan kepada penulis dalam menyelesaikan skripsi ini.
5. Herriyance S.T., M.Kom. selaku dosen pembimbing II yang telah memberikan bimbingan, kritik, motivasi, dan saran kepada penulis dalam menyelesaikan skripsi ini.
6. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer yang telah memberikan waktu dan tenaga untuk mengajar dan membimbing sehingga penulis dapat sampai kepada tahap penyusunan skripsi ini.
7. Seluruh Staf Pegawai Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah banyak memberikan bantuan kepada penulis selama masa perkuliahan sampai kepada tahap penyusunan skripsi ini.

8. Orang tua penulis khususnya Ibu yang telah memberikan motivasi, dukungan, doa serta saran dan kasih sayang penuh kepada penulis dalam menyelesaikan pendidikan.
9. Adik kandung tercinta Nazmi yang selalu mendukung serta mendoakan penulis dalam menjalankan aktivitas kuliah hingga akhirnya menyelesaikan tugas akhir.
10. Sahabat penulis Wilson, Anggi, Rezha, Mutia, Eca, Ulfa, dan Pahwana yang telah memberikan dukungan dan masukan positif kepada penulis.
11. Teman-teman seperjuangan dalam perkuliahan Sally, Sukiya, Susi, Chindy, Audrey, Thoriq, Andrew, Ariel, Avin, dan Imamul yang memberikan penulis semangat dan dukungan selama perkuliahan.
12. Seluruh staff pegawai PSI USU yang telah memberikan fasilitas dan dukungan untuk penulis selama mengerjakan skripsi.
13. Keluarga Besar IKLC USU yang telah memberikan dukungan, keceriaan dan pengalaman mengajar bersama yang berharga kepada penulis.
14. Teman-teman seluruh keluarga besar angkatan 2020 Ilmu Komputer Universitas Sumatera Utara yang telah banyak memberi motivasi kepada penulis.
15. Dan semua pihak yang telah membantu dan menyemangati yang tidak dapat disebutkan satu per satu.

Semoga anugerah Allah senantiasa menyertai setiap langkah dan memberikan keberkahan kepada semua pihak yang telah memberikan dorongan berupa semangat, arahan, dan dukungan baik secara kata maupun tindakan kepada penulis dalam menyelesaikan skripsi ini. Semoga hasil karya ini memberikan manfaat bagi individu, keluarga, masyarakat, organisasi, dan negara.

Medan, Desember 2023



Atikha Azmila
201401039

ABSTRAK

Dalam era modernisasi demokrasi, sistem e-voting atau pemungutan suara elektronik telah memunculkan berbagai tantangan dan peluang baru. Meskipun teknologi telah memfasilitasi kemudahan, efisiensi, dan aksesibilitas bagi pemilih, keamanan tetap menjadi perhatian utama. Salah satu isu krusial yang muncul dalam konteks ini adalah kerahasiaan suara. Di lingkungan pemilihan tradisional, kerahasiaan pemilih tetap terjaga karena suara dipilih secara anonim dan hanya hasil akhir yang diumumkan ke publik. Namun, dengan transisi ke dunia digital, menjaga kerahasiaan suara menjadi semakin rumit. Solusi inovatif diperlukan untuk mengatasi dilema ini.

Homomorphic Encryption, sebuah teknologi kriptografi asimetris, muncul sebagai solusi potensial. Dengan menggunakan Homomorphic Encryption, suara pemilih dapat dienkrpsi dan dihitung tanpa mengungkapkan preferensi individu. Setiap suara dienkrpsi sebelum dikirim ke sistem e-voting, memungkinkan otoritas pemilihan untuk menghitung total suara secara terenkrpsi. Dengan pendekatan ini, kerahasiaan suara dapat dipertahankan tanpa mengorbankan integritas pemilihan. Namun, memastikan bahwa suara dihitung dengan benar dan tidak dimanipulasi juga merupakan tantangan.

Menghadapi tantangan validasi suara, konsep Zero-Knowledge Proof muncul sebagai solusi efektif. Zero-Knowledge Proof memungkinkan pemilih untuk memverifikasi suara mereka tanpa mengungkapkan pilihan mereka. Dalam skenario ini, pemilih dapat memastikan bahwa suara mereka dihitung dengan benar tanpa kehilangan kerahasiaan. Dengan menggabungkan Homomorphic Encryption dan Zero-Knowledge Proof, sistem e-voting dapat mencapai tingkat keamanan dan kepercayaan yang tinggi.

Penerapan bersamaan Homomorphic Encryption dan Zero-Knowledge Proof menciptakan fondasi yang kuat untuk integritas dan kerahasiaan dalam sistem e-voting. Keberhasilan solusi ini tidak hanya memberikan jaminan kepada pemilih bahwa suara mereka tetap rahasia dan valid, tetapi juga meningkatkan kepercayaan publik terhadap proses demokrasi elektronik. Dengan demikian, e-voting bukan lagi sekadar mimpi teknologi, melainkan menjadi kenyataan yang aman dan andal untuk mendukung demokrasi modern.

Kata Kunci: *e-voting, Homomorphic Encryption, Paillier Cryptosystem, Zero-Knowledge Proof, Schnorr Protocol.*

TALLYING AND VALIDATION OF VOTES IN AN E-VOTING SYSTEM WITH HOMOMORPHIC ENCRYPTION AND ZERO-KNOWLEDGE PROOF PROTOCOL

ABSTRACT

In the era of democracy modernization, electronic voting systems, or e-voting, have introduced various new challenges and opportunities. While technology has facilitated convenience, efficiency, and accessibility for voters, security remains a paramount concern. One of the crucial issues in this context is the confidentiality of votes. In traditional voting environments, voter confidentiality is preserved as votes are cast anonymously, and only the final results are disclosed to the public. However, with the transition to the digital realm, maintaining the confidentiality of votes has become increasingly complex. Innovative solutions are needed to address this dilemma.

Homomorphic Encryption, an asymmetric cryptographic technology, has emerged as a potential solution. By utilizing Homomorphic Encryption, voters' choices can be encrypted and computed without revealing individual preferences. Each vote is encrypted before being transmitted to the e-voting system, allowing the election authorities to calculate the total votes in an encrypted form. With this approach, the confidentiality of votes can be maintained without compromising the integrity of the election. However, ensuring that votes are accurately counted and not manipulated poses a challenge.

To address the challenge of vote validation, the concept of Zero-Knowledge Proof has emerged as an effective solution. Zero-Knowledge Proof enables voters to verify their votes without disclosing their choices. In this scenario, voters can ensure that their votes are counted correctly without compromising confidentiality. By combining Homomorphic Encryption and Zero-Knowledge Proof, e-voting systems can achieve a high level of security and trust.

The simultaneous implementation of Homomorphic Encryption and Zero-Knowledge Proof creates a robust foundation for integrity and confidentiality in e-voting systems. The success of this solution not only provides assurance to voters that their votes remain confidential and valid but also enhances public trust in electronic democratic processes. Thus, e-voting is no longer just a technological dream but a secure and reliable reality supporting modern democracy.

Keywords: *e-voting, Homomorphic Encryption, Paillier Cryptosystem, Zero-Knowledge Proof, Schnorr Protocol.*

DAFTAR ISI

PERSETUJUAN	ii
PERNYATAAN.....	iii
PENGHARGAAN.....	iv
ABSTRAK	vi
ABSTRACT.....	viii
DAFTAR ISI.....	x
DAFTAR TABEL	xii
DAFTAR GAMBAR.....	xiii
 BAB I PENDAHULUAN.....	 1
1.1. Latar Belakang	1
1.2. Rumusan Masala	2
1.3. Batasan Masala	2
1.4. Tujuan Penelitian	3
1.5. Manfaat Penelitian	3
1.6. Metodologi Penelitian.....	3
1.7. Sistematika Penulisan	4
 BAB II LANDASAN TEORI	 5
2.1. Enkripsi	5
2.2. Homomorphic Encryption	9
2.3. Paillier Cryptosystem.....	10
2.4. Zero-Knowledge Proof	13
2.5. Schnorr Non-Interactive Zero-Knowledge Proof (Schnorr niZKP)	14
2.6. E-Voting.....	15
2.7. Penelitian yang Relevan.....	15
 BAB III ANALISIS DAN PERANCANGAN SISTEM.....	 17
3.1 Analisis Sistem.....	17
3.1.1 Analisis Masalah.....	17
3.1.2 Rancangan Kelas.....	18
3.1.3 Analisis Kebutuhan.....	18
3.1.4 Diagram Umum Sistem	19
3.2 Pemodelan Sistem.....	20
3.2.1 Flowchart	20
3.2.2 Pseudocode	24
 BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM	 28
4.1 Kompleksitas Algoritma	28
4.1.1 Kompleksitas Algoritma pada Fungsi random_odd_value.....	28
4.1.2 Kompleksitas Algoritma pada Fungsi is_prime.....	28
4.1.3 Kompleksitas Algoritma pada Fungsi generate_prime.....	29
4.1.4 Kompleksitas Algoritma pada Fungsi generate_keys.....	29
4.1.5 Kompleksitas Algoritma pada Fungsi find_gcd	30
4.1.6 Kompleksitas Algoritma pada Fungsi encrypt_message	31
4.1.7 Kompleksitas Algoritma pada Fungsi decrypt_message	31

4.1.8 Kompleksitas Algoritma pada Fungsi generate_proof	32
4.1.9 Kompleksitas Algoritma pada Fungsi verify_proof	33
4.1.10 Kompleksitas Algoritma pada Fungsi Utama	33
4.2 Pengujian Real Running Time pada Proses Perhitungan Suara Terenkripsi dan Dekripsi Total Suara	35
BAB V PENUTUP	40
5.1 Kesimpulan	40
5.2 Saran	40
DAFTAR PUSTAKA	41
LAMPIRAN	42

DAFTAR TABEL

Tabel 4.1 Tabel Kompleksitas Algoritma Fungsi random_odd_value	28
Tabel 4.2 Tabel Kompleksitas Algoritma Fungsi is_prime	28
Tabel 4.3 Tabel Kompleksitas Algoritma Fungsi generate_prime	29
Tabel 4.4 Tabel Kompleksitas Algoritma Fungsi generate_keys	29
Tabel 4.5 Tabel Kompleksitas Algoritma Fungsi find_gcd	30
Tabel 4.6 Tabel Kompleksitas Algoritma Fungsi encrypt_message	31
Tabel 4.7 Tabel Kompleksitas Algoritma Fungsi decrypt_message	31
Tabel 4.8 Tabel Kompleksitas Algoritma Fungsi generate_proof	32
Tabel 4.9 Tabel Kompleksitas Algoritma Fungsi verify_proof	33
Tabel 4.10 Tabel Kompleksitas Algoritma Fungsi utama	33
Tabel 4.11 Tabel Pengujian Real Running Time	35

DAFTAR GAMBAR

Gambar 2.1 Enkripsi Simetris	6
Gambar 2.2 Enkripsi Asimetris dengan Public key	8
Gambar 2.3 nkripsi Asimetris dengan Private key	8
Gambar 3.1 Ishikawa Diagram	17
Gambar 3.2 Class Diagram	18
Gambar 3.3 Diagram Umum Sistem.....	19
Gambar 3.4 Flowchart Sistem.....	21
Gambar 3.5 Flowchart Proses Validasi Suara.....	21
Gambar 3.6 Flowchart Proses Enkripsi Suara	22
Gambar 3.7 Flowchart Proses Penghitungan dan Dekripsi Total Suara.....	23
Gambar 4.1 Grafik Pengujian Rata-Rata Waktu Dengan Banyak Suara yang Masuk ...	35
Gambar 4.2 Grafik pengujian waktu percobaan 1 dengan banyak suara yang masuk....	36
Gambar 4.3 Grafik pengujian waktu percobaan 2 dengan banyak suara yang masuk....	37
Gambar 4.4 Grafik pengujian waktu percobaan 3 dengan banyak suara yang masuk....	37
Gambar 4.5 Grafik pengujian waktu percobaan 4 dengan banyak suara yang masuk....	38
Gambar 4.6 Grafik pengujian waktu percobaan 5 dengan banyak suara yang masuk....	39

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sistem e-voting atau pemungutan suara elektronik telah menjadi topik penting dalam upaya modernisasi proses demokrasi. Dengan memanfaatkan teknologi, e-voting dapat memberikan kemudahan, efisiensi, dan aksesibilitas yang lebih besar bagi pemilih. Namun, keamanan sistem e-voting tetap menjadi perhatian utama. Melindungi kerahasiaan suara dan mencegah manipulasi atau pemalsuan hasil pemilihan adalah faktor penting dalam memastikan integritas proses demokrasi.

Salah satu masalah utama yang dihadapi dalam sistem e-voting adalah kerahasiaan suara. Dalam sistem pemilihan suara tradisional, pemilih dapat memberikan suara mereka secara anonim dan hanya penghitungan suara yang terbuka untuk umum. Namun, di dunia digital, mengamankan kerahasiaan suara menjadi lebih rumit (Bharati et al., 2020). Pemilih harus dapat memberikan suara mereka tanpa mengkhawatirkan identitas mereka terungkap, sementara kalkulasi hasil penghitungan harus dilakukan tanpa melihat pilihan dari pemilih.

Untuk mengatasi masalah kerahasiaan suara dalam sistem e-voting, dapat digunakan Homomorphic Encryption. Homomorphic Encryption adalah sebuah skema kriptografi asimetris yang memungkinkan operasi matematika sederhana seperti penjumlahan dan perkalian dilakukan pada data terenkripsi. Dengan menggunakan Homomorphic Encryption, pihak otoritas e-voting dapat menghitung total suara yang terenkripsi tanpa mengungkapkan suara individu (Saksham, et al., 2020). Dalam skema ini, setiap suara dienkripsi dan dikirim ke sistem e-voting, kemudian dijumlahkan secara terenkripsi untuk mendapatkan hasil pemilihan tanpa mengorbankan kerahasiaan suara individu. Dengan demikian, Homomorphic Encryption dapat memberikan peningkatan keamanan dalam sistem e-voting.

Meskipun penerapan Homomorphic Encryption dapat memastikan kerahasiaan suara dalam sistem e-voting, namun ada kebutuhan untuk memastikan bahwa suara yang dihitung adalah sah dan tidak dimanipulasi. Untuk mengatasi masalah validasi suara, dapat digunakan protokol Zero-Knowledge Proof. Zero-Knowledge Proof adalah mekanisme kriptografi yang memungkinkan pemilih untuk memverifikasi

bahwa suara mereka telah dihitung dengan benar tanpa mengungkapkan pilihan dari pemilih.

Tanpa penerapan Zero-Knowledge Proof, pemilih mungkin merasa tidak yakin apakah suara mereka tidak dimanipulasi mengingat mereka tidak memiliki cara untuk memastikan tanpa mengorbankan kerahasiaan pilihan. Selain itu tanpa penerapan Zero-Knowledge Proof resiko terhadap penyalahgunaan suara oleh pihak lain juga dapat meningkat.

Dengan penerapan Homomorphic Encryption dan Zero-Knowledge Proof secara bersamaan dalam sistem e-voting, maka kepercayaan publik terhadap integritas dan kerahasiaan suara dapat ditingkatkan.

1.2. Rumusan Masalah

Rumusan masalah pada penelitian ini adalah melihat apakah *Homomorphic Encryption* dan protokol *Zero-Knowledge Proofs* membantu memvalidasi suara dan menghitung total suara terenkripsi secara efisien tanpa mengungkapkan suara individu dalam sistem e-voting.

1.3. Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Tidak membahas masalah keamanan lainnya seperti autentikasi pengguna maupun serangan jaringan.
2. Khusus membahas masalah kerahasiaan suara dalam sistem *e-voting* dan solusi yang diusulkan menggunakan *Homomorphic Encryption* dan protokol *Zero-Knowledge Proofs*.
3. Bahasa yang digunakan untuk membangun program adalah bahasa pemrograman *python*.
4. Tidak membahas isu sosial atau politik terkait *e-voting* seperti masalah kepercayaan publik, regulasi hukum, atau perspektif masyarakat terhadap *e-voting*.

1.4. Tujuan Penelitian

Tujuan yang diharapkan dari penelitian ini adalah:

1. Menghitung total suara terenkripsi secara efisien tanpa mengungkapkan suara individu dalam sistem *e-voting* menggunakan *Homomorphic Encryption*
2. Melakukan validasi suara dengan menggunakan *Zero-Knowledge Proofs* dalam sistem *e-voting*.

1.5. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah pengimplementasian *Homomorphic Encryption* dan protokol *Zero-Knowledge Proofs* dapat dijadikan pertimbangan untuk meningkatkan keamanan dalam pemilihan suara.

1.6. Metodologi Penelitian

Metode penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Pustaka

Pada tahap ini, awal penelitian dilakukan dengan mencari referensi dari berbagai sumber yang dapat dipercaya. Peninjauan pustaka dilaksanakan melalui studi buku, jurnal, *e-book*, artikel ilmiah, makalah, dan juga situs internet yang relevan dengan topik *Homomorphic Encryption* dan *Zero-Knowledge Proof*.

2. Analisis dan Perancangan

Berdasarkan cakupan penelitian, peneliti menganalisis elemen-elemen yang diperlukan untuk segera direncanakan dalam suatu diagram alir (*flowchart*).

3. Implementasi

Pada tahap ini, membuat sebuah sistem dengan menggunakan bahasa pemrograman *python* sesuai dengan diagram alir yang telah dirancang.

4. Pengujian

Pada tahap ini, dilakukan uji coba untuk memastikan bahwa sistem *e-voting* yang diimplementasikan dengan menggunakan *Homomorphic Encryption* dan *Zero-Knowledge Proof* berfungsi dengan baik sesuai dengan kebutuhan yang ditentukan.

5. Dokumentasi

Pada tahap ini, penelitian yang telah dilakukan, didokumentasikan mulai dari proses analisis hingga pengujian, dan disusun dalam format skripsi.

1.7. Sistematika Penulisan

Sistematika penulisan dari skripsi ini terdiri dari lima bab, yakni:

BAB I PENDAHULUAN

Segala aspek terkait dengan konteks penelitian, termasuk perumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, metode penelitian, dan struktur penulisan, diuraikan secara komprehensif dalam bab ini.

BAB II LANDASAN TEORI

Tinjauan teoritis yang berkaitan dengan *Homomorphic Encryption* dan protokol *Zero-Knowledge Proof* dibahas pada bab ini.

BAB III ANALISIS DAN PERANCANGAN

Analisis masalah dan sistem yang dibangun dibahas pada bab ini, lalu dilanjutkan dengan tahapan perancangan sistem dengan menggunakan *Homomorphic Encryption* dan protokol *Zero-Knowledge Proof*.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini berisi implementasi dan pengujian sistem yang didasarkan dari tahapan analisis dan perancangan.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari penelitian yang sudah dilaksanakan dan saran untuk penelitian selanjutnya.

BAB II

LANDASAN TEORI

2.1. Enkripsi

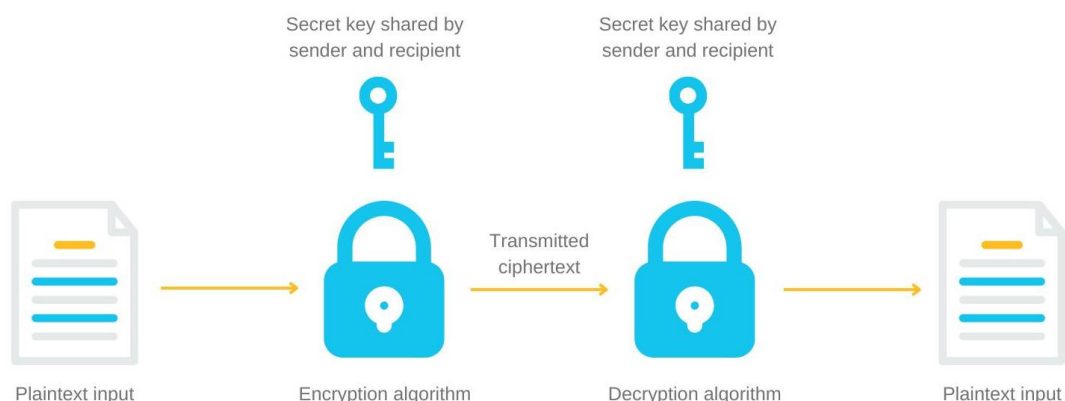
Enkripsi adalah pengaplikasian utama dari kriptografi, hal ini membuat data menjadi sulit dimengerti untuk menjaga kerahasiaannya. Enkripsi menggunakan sebuah algoritma yang disebut *cipher* dan nilai rahasia yang disebut dengan *key*. Saat kita melakukan enkripsi pesan, *plaintext* merujuk pada pesan yang belum dienkripsi, sedangkan *ciphertext* merujuk pada pesan yang telah dienkripsi. Sebuah *cipher* terdiri dari dua fungsi: enkripsi yang mengubah *plaintext* menjadi *ciphertext*, dan dekripsi yang mengubah kembali *ciphertext* menjadi *plaintext* (Aumasson, 2018). Terdapat dua jenis enkripsi utama, yaitu enkripsi simetris dan asimetris.

1. Enkripsi Simetris

Pada enkripsi simetris, *key* yang digunakan untuk mendekripsi adalah *key* yang sama untuk melakukan enkripsi (tidak seperti enkripsi asimetris atau public-key encryption, dimana *key* yang digunakan untuk melakukan dekripsi berbeda dengan *key* yang digunakan untuk melakukan enkripsi) (Aumasson, 2018). Skema enkripsi simetris memiliki lima komponen utama (Gambar 8.1.1):

- a. *Plaintext*: Pesan atau data yang dapat dibaca yang diterima oleh algoritma sebagai masukan.
- b. Algoritma Enkripsi: Algoritma enkripsi adalah suatu metode yang melakukan serangkaian transformasi pada teks asli (*plaintext*).
- c. *Secret Key*: *Secret key* juga menjadi *input* untuk algoritma enkripsi. Kunci adalah nilai yang tidak tergantung pada teks biasa (*plaintext*) dan metode algoritma. Keluaran dari algoritma akan bervariasi tergantung pada kunci yang digunakan. Proses substitusi dan transformasi yang dilakukan oleh algoritma bergantung secara spesifik pada nilai kunci tersebut.
- d. *Ciphertext*: Hasil atau *output* berupa pesan atau data acak yang hasilnya bergantung pada *plaintext* dan *secret key*. Untuk sebuah pesan, dua *key* yang berbeda dapat menghasilkan dua *ciphertexts* yang berbeda.

- e. Algoritma deskripsi: Pada dasarnya adalah algoritma enkripsi yang dijalankan secara terbalik. Algoritma ini menggunakan *ciphertext* dan *secret key* untuk menghasilkan *plaintext* (Stallings, 2022).



Gambar 2.1 Enkripsi Simetris

2. Enkripsi Asimetris

Enkripsi asimetris adalah enkripsi dimana *key* yang digunakan untuk melakukan dekripsi berbeda dengan *key* yang digunakan untuk melakukan enkripsi, karena itu pada enkripsi asimetris terdapat dua *key*. *Key* yang digunakan untuk melakukan enkripsi disebut *public key* dan secara umum tersedia untuk siapa saja yang mengirimkan pesan yang sudah terenkripsi. *Key* yang digunakan untuk melakukan dekripsi harus tetap dirahasiakan dan disebut sebagai *private key*.

Public key dapat dengan mudah dikomputasi dari private key, tetapi tentu saja demi keamanan dari serangan hacker private key tidak boleh dengan mudah dikomputasi dari public key. Dengan kata lain, mudah untuk melakukan komputasi satu arah, namun tidak ke arah sebaliknya—dan itu adalah tujuan dari public key kriptografi, dimana fungsi fungsi tersebut mudah di komputasi ke satu arah namun secara praktikal tidak dapat dibalik.

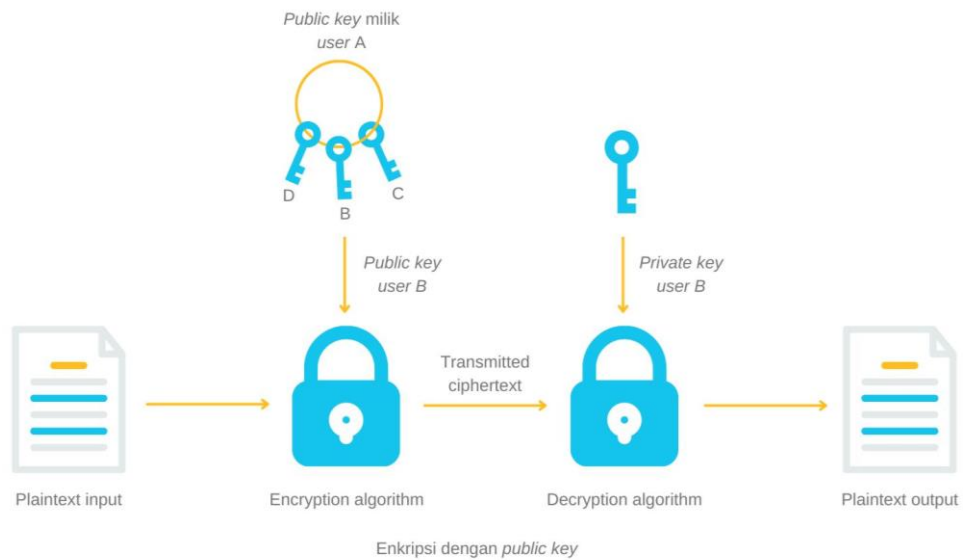
Model serangan (attack models) dan tujuan keamanan (security goals) untuk enkripsi asimetris hampir sama dengan enkripsi simetris, kecuali karena *key* yang digunakan untuk melakukan enkripsi bersifat public, setiap penyerang dapat membuat queries dari enkripsi dengan menggunakan public key untuk melakukan enkripsi. Model default untuk enkripsi asimetris adalah chosen-plaintext attacker (CPA) (Aumasson, 2018).

Enkripsi asimetris memiliki enam komponen:

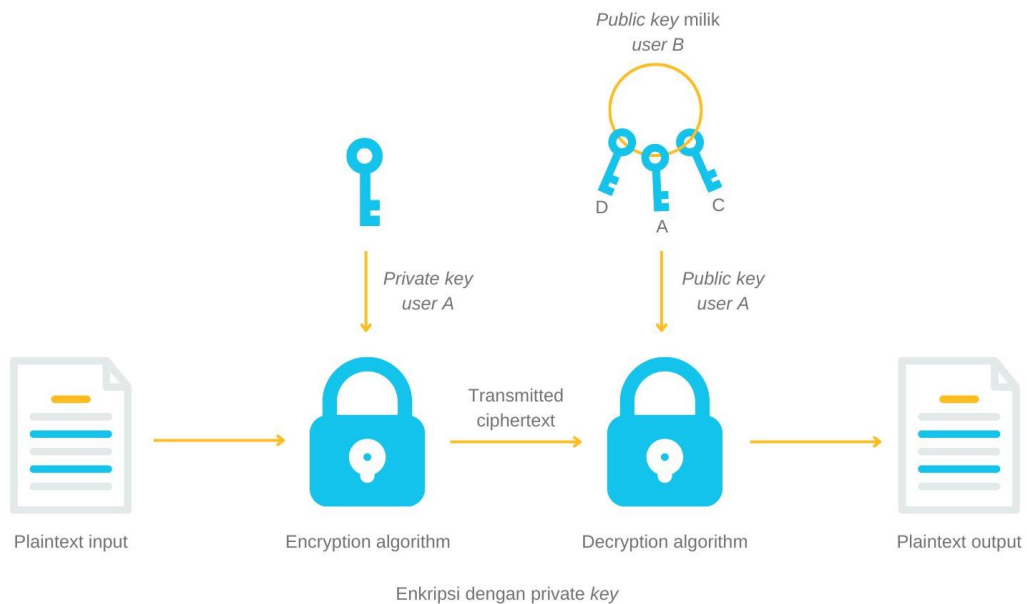
- a. *Plaintext*: Pesan atau data yang dapat dibaca yang diterima oleh algoritma sebagai masukan.
- b. Algoritma Enkripsi: Algoritma enkripsi adalah suatu metode yang melakukan serangkaian transformasi pada teks asli (*plaintext*).
- c. *Public dan private keys*: Sepasang kunci yang telah dipilih sedemikian rupa sehingga jika salah satu digunakan untuk melakukan enkripsi, yang lainnya digunakan untuk melakukan dekripsi. Transformasi yang dilakukan oleh algoritma bergantung pada *public key* atau *private key* yang disediakan sebagai *input*.
- d. *Ciphertext*: Hasil atau output berupa pesan atau data acak yang hasilnya bergantung pada plaintext dan key. Untuk sebuah pesan, dua key yang berbeda dapat menghasilkan dua ciphertexts yang berbeda.
- e. Algoritma dekripsi: Algoritma yang menerima ciphertext dan key yang sesuai untuk menghasilkan plaintext.

Langkah-langkah utamanya adalah sebagai berikut:

1. Setiap pengguna membuat sepasang *key* untuk digunakan dalam melakukan enkripsi dan dekripsi pesan.
2. Setiap pengguna menggunakan salah satu dari kedua jenis *keys* kedalam *public registry* atau *file* lain yang dapat diakses. Seperti pada Gambar 8.1.2 dan Gambar 8.1.3, setiap pengguna menyimpan *public keys* yang diperoleh pengguna lain.
3. Jika *user A* ingin mengirimkan pesan rahasia kepada *user B*, *user A* mengenkripsi pesannya menggunakan *public key* milik *user B*.
4. Ketika *user B* menerima pesannya, ia melakukan dekripsi menggunakan *private key* miliknya. Tidak ada orang lain yang dapat mendekripsi pesannya karena hanya *user B* yang mengetahui *private key* miliknya.



Gambar 2.2 Enkripsi Asimetris dengan *Public key*



Gambar 2.3 Enkripsi Asimetris dengan *Private key*

Dengan pendekatan ini, semua pengguna memiliki akses memiliki akses ke *public key*, dan *private key* dihasilkan secara lokal oleh masing - masing pengguna dan oleh karena itu tidak perlu didistribusikan. Selama *private key* pengguna tetap terlindungi dan dirahasiakan, komunikasi yang masuk akan aman. Kapan pun, sistem dapat mengganti *private key*-nya dan memberikan *public key* pendamping untuk menggantikan *public key* lainnya (Stallings, 2022).

2.2. Homomorphic Encryption

Homomorphic encryption adalah suatu metode enkripsi yang memungkinkan komputasi dilakukan pada data yang telah dienkripsi atau *ciphertext*. Secara matematis, *homomorphic encryption* melakukan transformasi dari satu set data ke set data lain sambil tetap mempertahankan relasi yang ada pada set data asli. Umumnya, skema *homomorphic encryption* efektif bekerja dengan data yang direpresentasikan sebagai bilangan bulat. Terdapat dua jenis operasi dalam *homomorphic encryption*, yaitu operasi aditif dan operasi multiplikatif. Selain itu, terdapat tiga jenis homomorphic encryption yang berbeda, yaitu:

1. *Partially homomorphic encryption* (PHE)

PHE memungkinkan untuk dilakukan satu operasi tertentu untuk dilakukan pada pesan terenkripsi.

2. *Somewhat homomorphic encryption* (SHE)

SHE memungkinkan eksekusi dua operasi khusus dengan jumlah operasi yang terbatas.

3. *Fully homomorphic encryption* (FHE)

FHE memungkinkan untuk dilakukannya dua operasi tertentu pada pesan terenkripsi.

Dalam konteks aljabar abstrak, homomorfisme merujuk pada suatu pemetaan yang memelihara struktur antara dua struktur aljabar, seperti grup. Grup sendiri merupakan himpunan, G , bersama dengan operasi \circ (disebut dengan hukum grup G) yang menggabungkan dua elemen a dan b untuk membentuk elemen lain, dilambangkan dengan ab . Untuk memenuhi syarat sebagai grup, himpunan dan operasi, $(G; \circ)$ harus memenuhi empat persyaratan yang dikenal sebagai aksioma grup:

- *Closure*: Untuk semua, a, b di G , hasil dari operasi $a \circ b$, juga ada di G
- *Associativity*: Untuk semua, a, b , dan c pada G , $(a \circ b) \circ c = a \circ (b \circ c)$
- *Elemen identitas*: Terdapat elemen e di G , sehingga untuk setiap elemen a di G , berlaku persamaan $e \circ a = a \circ e$.
- *Elemen inverse*: Untuk setiap a di G , terdapat elemen b di G sehingga $a \circ b = b \circ a = e$, dimana e adalah elemen identitas.

Elemen identitas dari sebuah himpunan G biasa ditulis sebagai 1. Persamaan $ab = ba$ selalu berlaku dalam himpunan bilangan bulat dengan operasi penjumlahan $a + b = b + a$ untuk setiap dua bilangan bulat (komutativitas penjumlahan).

Himpunan yang selalu memenuhi persamaan komutativitas $ab = ba$ disebut himpunan abelian.

Diberikan dua himpunan $(G;)$ dan $(H;)$, sebuah himpunan homomorfisme dari $(G;)$ ke $(H;)$ adalah sebuah fungsi $f : G \rightarrow H$ sehingga untuk setiap g dan g' di G berlaku:

$$f(g \diamond g') = f(g) \circ f(g')$$

Diberikan (P, C, K, E, D) sebagai sebuah skema enkripsi, dimana P, C adalah plaintext dan ciphertext, K adalah key, dan E, D adalah algoritma enkripsi dan dekripsi. Diasumsikan bahwa plaintext membentuk sebuah himpunan $(P,)$ dan ciphertext membentuk sebuah himpunan $(C,)$, maka algoritma enkripsi E adalah pemetaan dari himpunan P ke himpunan C , yaitu $E_k : P \rightarrow C$, dimana $k \in K$ adalah sebuah secret key atau sebuah public key.

Untuk semua a dan b di P dan k di K , jika

$$E_k(a) \circ E_k(b) = E_k(a \diamond b)$$

Skema enkripsinya adalah **homomorphic** (Yi, *et al.*, 2014).

2.3. Paillier Cryptosystem

Paillier Cryptosystem adalah sebuah sistem kriptografi *homomorphic* aditif, yang berarti hasil perkalian dari dua *ciphertext* dapat dihitung tanpa harus melakukan dekripsi, dan hasilnya merupakan *ciphertext* dari hasil perkalian *plaintext* (Katz, 2015). Pada tahun 1999, Pascal Paillier mengembangkan *Paillier cryptosystem* sebagai suatu sistem kriptografi kunci publik yang bersifat probabilistik. Dalam algoritma *Paillier*, enkripsi data dilakukan menggunakan kunci publik, sementara dekripsi data menggunakan kunci privat.

Algoritma pembentukan *key* dari *Paillier* adalah sebagai berikut:

1. Tentukan dua buah bilangan prima p dan q secara acak, sehingga memenuhi $\text{fpb}(pq, (p-1)(q-1)) = 1$. Disini, FPB merujuk pada Faktor Persekutuan Terbesar. Kondisi persamaan tersebut dipastikan terpenuhi apabila panjang p dan q memiliki nilai yang sama.
2. Hitung $n = pq$ dan $\lambda = \text{kpk}(p-1, q-1)$. Disini kpk adalah kelipatan persekutuan terkecil.
3. Pilih sembarang bilangan bulat g , yang mana $g \in \mathbb{Z}_{n^2}^*$

- Pastikan bahwa order dari g dapat dibagi habis oleh n dengan melakukan verifikasi menggunakan rumus $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$, dimana fungsi L adalah $L(x) = \frac{x-1}{n}$.

- Public key* berupa (n, g) dan *private key* nya adalah (λ, μ) .

Algoritma untuk melakukan enkripsi adalah sebagai berikut.

- Contohnya, ada sebuah *plaintext* m yang akan dienkripsi, dengan batasan bahwa nilai pesannya berada dalam rentang antara 0 hingga n .
- Tentukan bilangan bulat r secara acak, yang mana $0 < r < n$ dan $\text{fpb}(r, n) = 1$
- Ciphertext* c dapat dihitung dengan rumus $c = g^m * r^n \bmod n^2$.

Algoritma untuk melakukan dekripsi adalah sebagai berikut.

- Misalkan c merupakan *ciphertext*, yang mana $c \in Z_{n^2}^*$.
- Plaintext* m dapat dihitung dengan rumus $m = (L(c^\lambda \bmod n^2) * \mu) \bmod n$

Sistem kriptografi *Paillier* memiliki karakteristik sebagai enkripsi homomorfik parsial yang mendukung operasi penjumlahan. Dengan sifat ini, dapat dirumuskan identitas berikut.

- Penjumlahan dari *plaintext*

Perkalian dari dua buah *ciphertext* akan menghasilkan penjumlahan dari *plaintext*-nya.

$$D(E(m_1, r_1) * E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n$$

- Perkalian dari *plaintext*

Dekripsi dari *ciphertext* yang dipangkatkan dengan *plaintext* akan menghasilkan perkalian kedua *plaintext*.

$$D(E(m_1, r_1)^{m_2} \bmod n^2) = m_1 * m_2 \bmod n$$

Sebuah contoh dari skema *Paillier encryption* dengan parameter kecil dapat dijelaskan sebagai berikut.

Untuk mempermudah perhitungan, contoh berikut akan memakai bilangan prima kecil untuk membuat nilai n yang kecil,

$$p = 13, q = 11$$

kemudian

$$n = p \cdot q = 13 \cdot 11 = 143$$

Selanjutnya, sebuah integer g harus dipilih dari Z_n^* . Jika kita memilih integer secara acak,

$$g = 144$$

jika semua properti yang diperlukan, maka *public key*-nya adalah,

$$(n, g) = (143, 144)$$

Untuk melakukan enkripsi pesan

$$m = 18$$

dimana $m \in Z_n$, pilih secara acak

$$r = 14$$

dimana r merupakan bilangan bulat bukan nol, dan $r \in Z_n$

Hitung

$$\begin{aligned} c &= g^m r^n \pmod{n^2} \\ &= 144^{18} 14^{143} \pmod{20449} \\ &= 2575 \cdot 9465 \pmod{20449} \\ &= 17616 \end{aligned}$$

Untuk melakukan dekripsi *ciphertext* c , hitung

$$\lambda = \text{lcm}(12, 10) = 60$$

Tentukan $L(u) = (u - 1) / n$, hitung

$$\begin{aligned} k &= L(g^\lambda \pmod{n^2}) \\ &= L(144^{60} \pmod{20449}) \\ &= L(8581) \end{aligned}$$

$$L(u) = u - 1/n = 8581 - 1/143 = 8580/143 = 60$$

Hitung inverse dari k ,

$$\begin{aligned} \mu &= k^{-1} \pmod{n} \\ &= 60^{-1} \pmod{143} = 31 \end{aligned}$$

Hitung

$$\begin{aligned} m &= L(c^\lambda \pmod{n^2}) \cdot \mu \pmod{n} \\ &= L(17616^{60} \pmod{20449}) \cdot 31 \pmod{143} \\ &= L(11298) \cdot 31 \pmod{143} \\ &= 79 \cdot 31 \pmod{143} = 18 \end{aligned}$$

2.4. Zero-Knowledge Proof

Zero-Knowledge Proof (ZKP) adalah salah satu protokol kriptografi yang memungkinkan seorang *prover* untuk membuktikan kepemilikan informasi rahasia kepada pihak lain yang biasa disebut *verifier* tanpa harus mengungkapkan detail informasi tersebut atau memberikan cara bagi pihak lain untuk mengakses rahasia tersebut.

1. *Prover dan Identifier (P, V)*

Prover (pemberi bukti) adalah entitas yang memberikan bukti. Sebaliknya, pengertian verifier (verifikator) umumnya fokus pada proses verifikasi atau peran dari verifier. Ketidakseimbangan antara kompleksitas dari verifikasi dan teorema pembuktian masuk ke dalam cakupan kelas kompleksitas NP, yang dapat dianggap sebagai sebuah kelas dari proof systems.

2. *Properti Protokol Zero-Knowledge Proof*

Terdapat dua properti utama yang harus dipenuhi oleh sebuah proof system (prosedur verifikasi), yaitu soundness (atau validitas) dan completeness. Soundness menyatakan bahwa prosedur verifikasi tidak dapat “dibohongi” untuk menerima pernyataan palsu. Dengan kata lain, soundness menggambarkan kemampuan verifier untuk melindungi dirinya agar tidak dipengaruhi oleh pernyataan palsu. Di sisi lain, completeness menggambarkan kemampuan dari prover untuk meyakinkan verifier tentang pernyataan yang benar. Mengingat bahwa Zero-Knowledge Proof adalah prosedur verifikasi dimana verifier tidak mempelajari apapun (tidak memperoleh “pengetahuan”), protokol Zero-Knowledge Proof harus memenuhi tiga properti, yaitu completeness, soundness, dan zero-knowledge.

3. *Interactive Proof Systems*

\Pengertian dari Interactive Proof Systems merujuk pada dua aktivitas komputasi yang berkaitan dengan proof system, yaitu menghasilkan sebuah bukti dan memverifikasi validitas dari bukti tersebut. Dua aktivitas tersebut dilakukan oleh dua pihak yang berbeda, yaitu prover dan verifier, yang berinteraksi dengan satu sama lain. Dalam beberapa kasus, interaksi ini bisa sangat sederhana dan hanya bergerak satu arah (prover mengirimkan sebuah teks sebagai bukti, kepada verifier) (Goldreich, 2001).

4. *Non-Interactive Zero-Knowledge Proofs (niZKP)*

Pada non-interactive zero-knowledge proof adalah ZKP yang tidak memerlukan interaksi antara prover dan verifier. Sebaliknya, prover membuat sebuah bukti yang dapat di verifikasi oleh verifier tanpa adanya interaksi.

2.5. Schnorr Non-Interactive Zero-Knowledge Proof (Schnorr niZKP)

Schnorr non-interactive zero-knowledge (NIZK) proof adalah varian non-interaktif dari skema identifikasi *three-pass Schnorr*. *Schnorr NIZK proof* memungkinkan seseorang untuk membuktikan pengetahuan tentang logaritma diskrit tanpa membocorkan informasi apapun tentang isinya. *Schnorr NIZK proof* yang digunakan adalah *Schnorr NIZK proof* atas *finite-field* yang berjalan secara interaktif antara *prover* (Alice) dan *verifier* (Bob). Dalam skema ini, Alice mempublikasikan *public-key* nya $A = g^a \bmod p$, dimana a adalah *private-key* yang dipilih secara acak dari rentang $[0, q - 1]$.

Protokol ini bekerja dalam *three-passes* (tiga langkah):

1. Alice memilih sebuah angka v secara acak dari rentang $[0, q - 1]$ dan menghitung $V = g^v \bmod p$. Kemudian dia mengirimkan V ke Bob.
2. Bob memilih sebuah *challenge* c secara acak dari rentang $[0, 2^t - 1]$, di mana t adalah panjang bit dari *challenge* tersebut (misalnya, $t = 160$) kemudian, Bob mengirimkan nilai c kepada Alice.
3. Alice melakukan perhitungan $r = v - a * c \bmod q$ dan mengirimkannya ke Bob.

Pada akhir protokol tersebut, Bob melakukan beberapa pemeriksaan berikut. Jika ada salah satu pemeriksaan yang gagal, maka identifikasi tidak berhasil.

1. Untuk memverifikasi A dalam rentang $[1, p - 1]$ dan $A^q = 1 \bmod p$;
2. Untuk memverifikasi $V = g^r * A^c \bmod p$.

Schnorr NIZK proof diperoleh dari skema identifikasi *Schnorr* interaktif melalui transformasi *Fiat-Shamir*, dimana mengubah sebuah protokol interaktif menjadi non-interaktif (Fiat, 1986).

2.6. E-Voting

E-Voting, atau Pemungutan Suara Elektronik, merujuk pada penerapan teknologi informasi dalam proses penyelenggaraan pemilihan umum.

Dalam konteks pemilihan suara atau pemilihan umum, istilah "suara" merujuk pada hak atau tindakan memberikan dukungan atau preferensi terhadap kandidat, partai politik, atau pilihan lainnya. Suara dalam pemilihan umum sering kali diwakili oleh tindakan memilih atau memberikan suara secara resmi untuk mendukung calon atau opsi tertentu.

Pengimplementasian dan teknologi *e-voting* terus mengalami evolusi seiring dengan pesatnya perkembangan teknologi informasi. Kendala-kendala yang muncul dalam berbagai negara yang telah dan sedang menerapkan *e-voting* menjadi landasan untuk penyempurnaan sistem ini. Salah satu keuntungan dari penerapan *e-voting* saat ini adalah penurunan biaya perangkat keras dan keterbukaan perangkat lunak, yang membuat biaya pelaksanaan *e-voting* semakin terjangkau. Selain itu, perangkat lunak juga semakin terbuka untuk diaudit bersama.

Setelah putusan Mahkamah Konstitusi pada tanggal 30 Maret 2010 yang menyatakan bahwa penggunaan *e-voting* sesuai dengan konstitusi asalkan tetap mematuhi prinsip pemilu yang bebas dan adil, maka pelaksanaan *e-voting* dapat diperluas secara lebih luas.

2.7. Penelitian yang Relevan

Beberapa penelitian terdahulu yang relevan dengan penelitian ini yaitu tentang *Paillier cryptosystem*, *Schnorr non-interactive zero-knowledge proof*, dan *e-voting* :

1. Pada jurnal berjudul *Online Voting System Using Homomorphic Encryption* (Saproo, *et al.*, 2020). Penelitian yang dilakukan adalah membangun suatu sistem *e-voting* berbasis web yang mengimplementasikan *Paillier cryptosystem* untuk menjaga integritas data.
2. Pada jurnal berjudul *Secure E-Voting System Based on Paillier Cryptography* (Raut, *et al.*, 2020). Pada penelitian ini dapat ditarik kesimpulan bahwa sistem *e-voting* menggunakan *Paillier cryptosystem* adalah sistem yang fleksibel dan aman untuk digunakan. Penggunaan sistem ini juga dapat menghemat waktu dan mengurangi campur tangan manusia.

3. Pada jurnal berjudul *Schnorr non-interactive zero-knowledge proof* (Hao, 2017). Penelitian yang dilakukan adalah mengubah protokol *Schnorr* yang semula interaktif menjadi non-interaktif melalui transformasi *Fiat-Shamir*.

BAB III

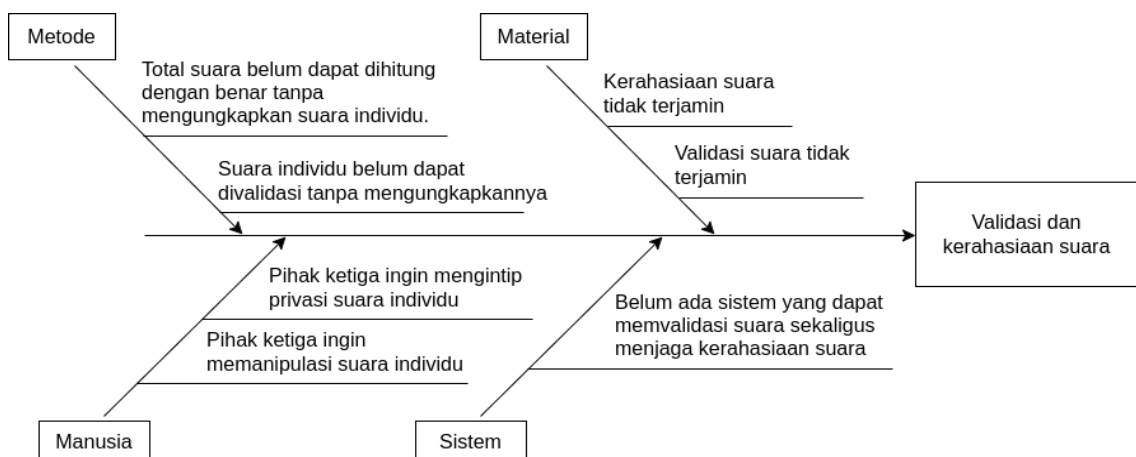
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Salah satu langkah penting dalam melakukan penelitian adalah analisis sistem. Analisis ini terdiri dari dua jenis, yaitu analisis masalah dan analisis kebutuhan, dimana analisis masalah membantu mengidentifikasi penyebab dan dampak dari suatu masalah, sementara analisis kebutuhan membantu menjelaskan data dan proses yang diperlukan untuk merancang sebuah sistem. Dengan kata lain, analisis sistem membantu memahami bagian-bagian yang diperlukan agar sistem dapat berjalan dengan baik.

3.1.1 Analisis Masalah

Pada tahap ini akan dilakukan identifikasi penyebab terjadinya masalah. Pada penelitian ini, masalah terdapat pada cara memvalidasi suara dan menghitung total suara terenkripsi secara efisien tanpa mengungkapkan suara individu pada sistem *e-voting*. Analisis masalah akan dilakukan melalui penerapan diagram *Ishikawa*, juga dikenal sebagai *Fishbone diagram*. Diagram *Ishikawa* memiliki struktur yang menyerupai tulang ikan, terdiri dari kepala ikan yang berisi judul atau tujuan permasalahan, dan tulang-tulang ikan sebagai penjelasan dari penyebab permasalahan. Uraian permasalahan pada penelitian ini dalam diagram *Ishikawa* pada gambar 3.1.

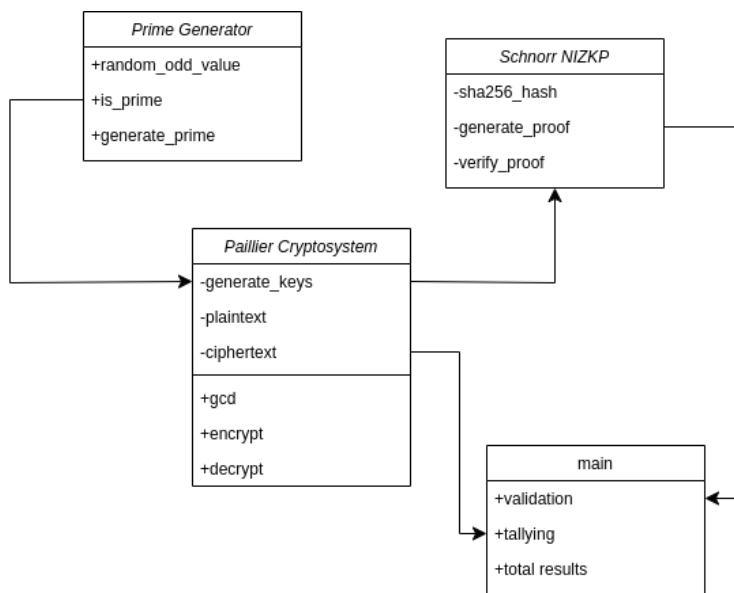


Gambar 3.1 Ishikawa Diagram

Diagram diatas menggambarkan akar penyebab dari permasalahan yang ingin diselesaikan dalam penelitian ini.

3.1.2 Rancangan Kelas

Komponen pada sistem ini beroperasi di sisi klien. Komponen tersebut bertanggung jawab untuk validasi suara, enkripsi suara, perhitungan total suara dalam bentuk terenkripsi, dan dekripsi total suara. Diagram kelas pada komponen tersebut adalah sebagai berikut:



Gambar 3.2 *Class Diagram*

3.1.3 Analisis Kebutuhan

Dalam analisis ini, dilakukan identifikasi kebutuhan yang dibagi menjadi dua kategori, yaitu kebutuhan fungsional dan kebutuhan non-fungsional, dimana kebutuhan fungsional mencakup aktivitas yang dapat dilakukan oleh sistem, sementara kebutuhan non-fungsional melibatkan aspek tambahan yang digunakan untuk menyempurnakan sistem, seperti fitur, karakteristik, dan batasan lainnya.

3.1.3.1 Kebutuhan Fungsional

Kebutuhan fungsional mencakup langkah-langkah atau proses yang harus dilakukan oleh sistem. Dalam konteks sistem ini, kebutuhan fungsional merinci proses-proses yang harus dijalankan. Kebutuhan fungsional yang harus dipenuhi pada sistem ini adalah:

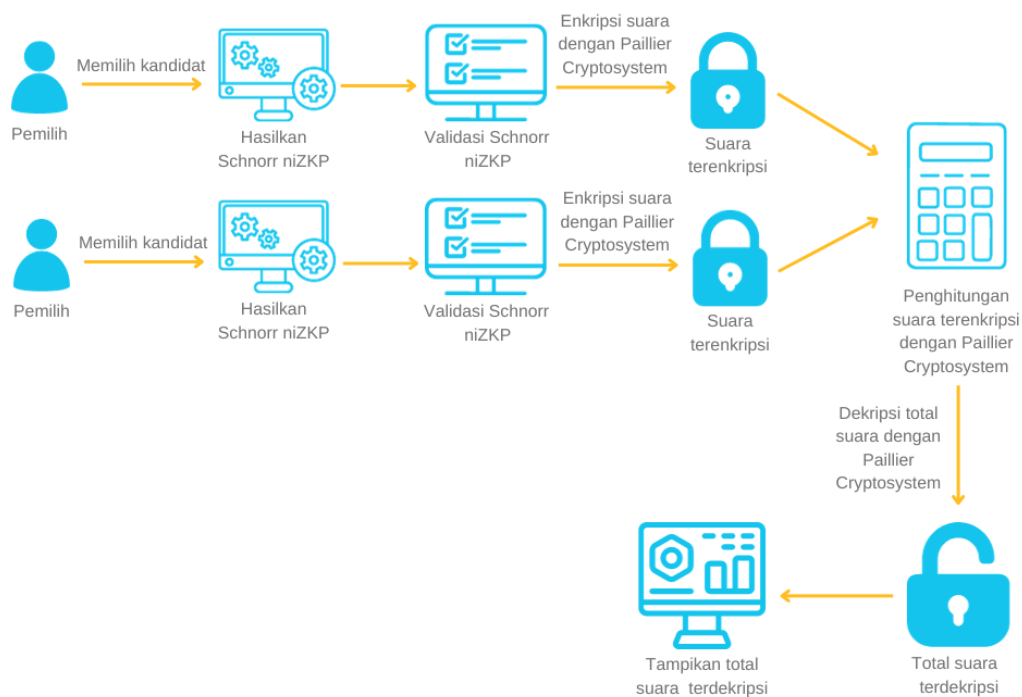
1. Sistem dapat melakukan validasi bahwa suara yang masuk ke dalam sistem adalah valid dan tidak dimanipulasi.
2. Sistem dapat melakukan enkripsi pada setiap suara individu.
3. Sistem dapat menghitung total suara dalam bentuk terenkripsi.
4. Sistem dapat melakukan dekripsi total suara dari setiap kandidat.
5. Sistem dapat melakukan dekripsi total suara dari semua kandidat.

3.1.3.1 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan kebutuhan yang digunakan dalam proses kerja sistem. Berikut adalah kebutuhan non-fungsional pada sistem ini:

1. Sistem dapat memberitahu kepada pengguna bahwa suara telah terenkripsi dan telah divalidasi
2. Sistem dapat menampilkan total suara terenkripsi dari setiap kandidat.
3. Sistem dapat menampilkan total suara terenkripsi dari kedua kandidat.
4. Sistem dapat menampilkan total suara terdekripsi dari setiap kandidat.
5. Sistem dapat menampilkan total suara terdekripsi dari kedua kandidat.

3.1.4 Diagram Umum Sistem



Gambar 3.3 Diagram Umum Sistem

Diagram umum sistem *e-voting* dengan *homomorphic encryption* dan *zero-knowledge proof* pada penelitian ini dimulai dengan pemilihan suara oleh pemilih. Setelah itu akan dihasilkan *Schnorr niZKP* (*non-interactive zero-knowledge proof*). *Schnorr niZKP* tersebut lalu akan divalidasi untuk melihat apakah suara telah dimanipulasi atau tidak. Jika suara valid, maka suara akan di enkripsi menggunakan *Paillier cryptosystem*. Suara terenkripsi tersebut akan dikumpulkan bersama suara terenkripsi lainnya dan akan dilakukan penghitungan kumpulan suara tersebut dalam bentuk terenkripsi menggunakan *Paillier cryptosystem*. Lalu total suara terenkripsi akan didekripsi menggunakan *Paillier cryptosystem*. Total suara terdekripsi akan ditampilkan untuk melihat hasil pemungutan suara.

3.2 Pemodelan Sistem

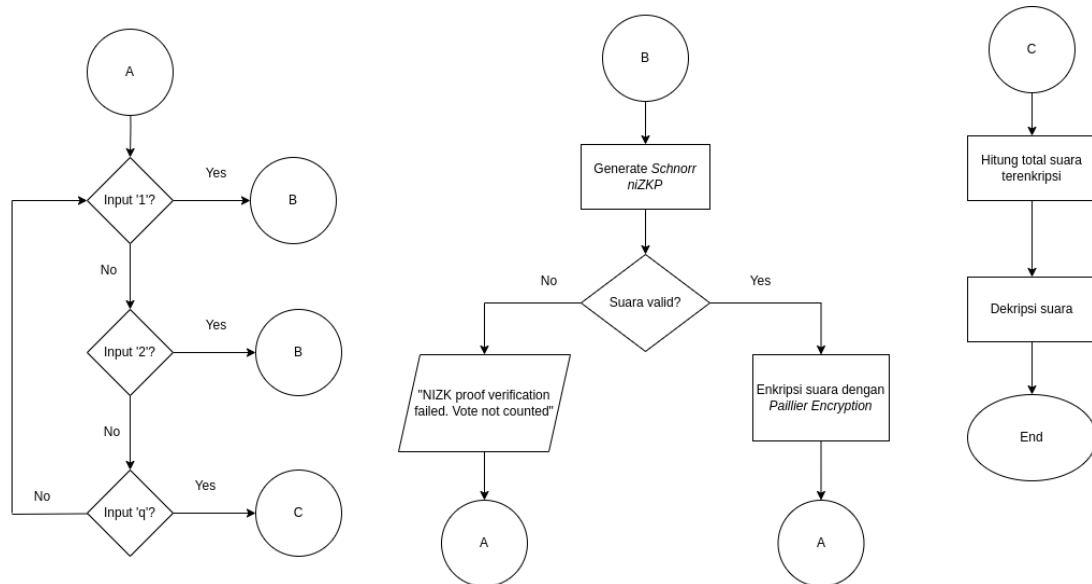
Pemodelan sistem merupakan representasi visual dari elemen-elemen yang diperlukan dalam suatu sistem yang sedang direncanakan. Dalam penelitian ini, sistem akan dijelaskan melalui *flowchart*.

3.2.1 Flowchart

Flowchart ialah representasi grafis yang mencakup zona-zona yang menggambarkan langkah-langkah dalam menyelesaikan suatu masalah. *Flowchart* berfungsi sebagai cara untuk memvisualisasikan suatu algoritma.

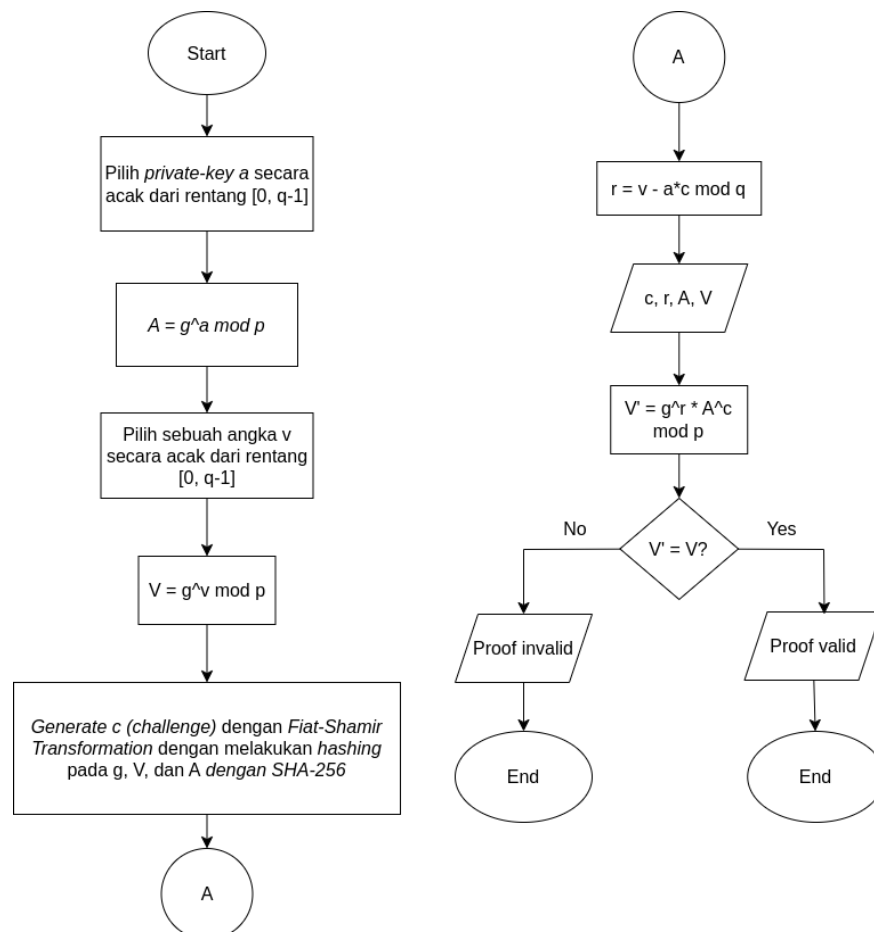
3.2.1.1 Flowchart Sistem

Flowchart sistem adalah representasi visual dari langkah-langkah yang terlibat dalam validasi suara dan pengambilan suara. Selanjutnya, akan diuraikan menjadi tiga bagian, yaitu *flowchart* validasi suara, *flowchart* pemilihan suara, dan *flowchart* penghitungan total suara.



Gambar 3.4 *Flowchart* Sistem

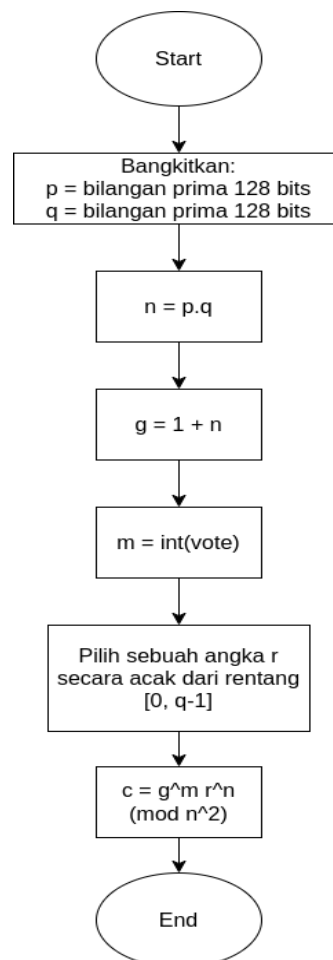
3.2.1.2 *Flowchart* Validasi Suara



Gambar 3.5 *Flowchart* Proses Validasi Suara

Berdasarkan gambar ..., dijelaskan bahwa untuk melakukan validasi suara, a sebagai *private-key* akan dihasilkan secara acak dengan rentang $[0, q - 1]$. Lalu dengan menggunakan a , akan dihitung *public-key* yaitu $A = ga \bmod p$. Setelah itu hasilkan angka v secara acak dengan rentang $[0, q - 1]$ dan hitung *commitment* dengan rumus $V = gv \bmod p$. Lalu hasilkan c (*challenge*) menggunakan transformasi *Fiat-Shamir* dengan melakukan *hashing string* yang terdiri dari g, V , dan A . Kemudian hitung *response* dengan rumus $r = v - a^c \bmod q$. *Proof* (c, r, A, V) akan dikirimkan. Pada langkah akhir protokol, akan dilakukan pengecekan. Yaitu menghitung $V' = gr \cdot A^c \bmod p$. Jika $V' = V$ maka suara valid dan siap untuk melakukan enkripsi. Jika suara tidak valid, maka suara yang dikirimkan tidak akan dihitung.

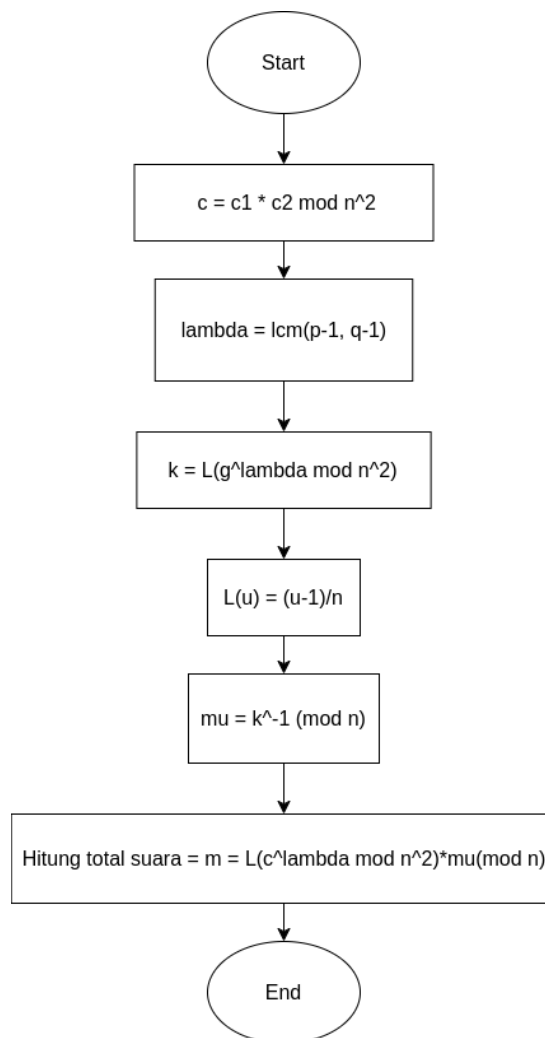
3.2.1.3 Flowchart Enkripsi Suara



Gambar 3.6 Flowchart Proses Enkripsi Suara

Pada proses enkripsi suara, langkah pertama yang dilakukan adalah menghasilkan p dan q yang merupakan bilangan prima 128 bits dengan syarat $p \neq q$. Lalu akan dihitung n yang merupakan *public-key* dengan rumus $n = p \cdot q$. Setelah itu akan dihasilkan *public-key* lain yaitu g dengan rumus $g = 1 + n$. Lalu pilih sebuah angka r secara acak dari rentang $[0, q - 1]$. Kemudian, g, m, r , dan n akan digunakan untuk melakukan enkripsi (menghasilkan *ciphertext*) dengan rumus $c = g^m \cdot r^n \pmod{n^2}$.

3.2.1.4 Flowchart Penghitungan Suara Terenkripsi dan Dekripsi Total Suara



Gambar 3.7 Flowchart Proses Penghitungan dan Dekripsi Total Suara

Perkalian dari dua buah ciphertext akan menghasilkan penjumlahan dari plaintext-nya. Karena itu akan dilakukan perkalian *ciphertext* dengan rumus $c = c_1 \cdot c_2 \bmod n^2$. Lalu akan dihasilkan sebuah *private-key* dengan rumus $\lambda = \text{lcm}(p-1, q-1)$. Kemudian, hitung k dengan rumus $k = L(g^\lambda \bmod n^2)$ dengan $L(u) = u - 1/n$. Setelah itu, cari μ dengan rumus $\mu = k^{-1} \bmod 77$. Lalu akan dilakukan dekripsi total suara dengan rumus $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$.

3.2.2 Pseudocode

Pseudocode adalah metode penulisan program yang bersifat informal dan dapat disusun dengan aturan yang ditetapkan sendiri. Dengan kata lain, pseudocode adalah urutan langkah logis yang dibuat agar mudah dipahami oleh manusia.

3.2.2.1 Pseudocode Prime Generator

Class PrimeGenerator:

```

Function random_odd_value(bits) {
    Choose a random  $n \in \mathbb{Z}$ ,  $2^{(\text{bits}-1)} \leq n < 2^{\text{bits}} - 1$ 
    If  $n$  is even, increment it by 1
    Return  $n$ 
}

Function is_prime( $n$ ,  $j=50$ ) {
    If  $n$  is less than or equal to 1, return False
    If  $n$  is less than or equal to 3, return True

    Initialize  $k$  to 0 and  $m$  to  $n - 1$ 
    While  $m$  is even {
        Increment  $k$  by 1
        Divide  $m$  by 2
    }

    Repeat the following  $j$  times {

```

```

Function generate_prime(bits) {
    Repeat indefinitely {
        Generate a random odd value n
        from random_odd_value function with the specified
        number of bits
    }
}

```

3.2.2.2 Pseudocode Paillier Cryptosystem

Class **Paillier**:

```

Function generate_keys() {
    Initialize bits to 128
    Create a PrimeGenerator instance
    Create a Paillier instance
    Create a SchnorrNIZKP instance
    Generate prime numbers p and q using the
PrimeGenerator, but p is not equal to q
    Calculate n as p * q
    Initialize g to 1 + n
    Calculate lambda as (p - 1) * (q - 1)
    Calculate k as the result of (g^lambda mod n^2) - 1
    / n
    Calculate mu as the modular inverse of k mod n
}

Function find_gcd(a, b) {
    While b is not zero {
        Set a to b
        Set b to the remainder when a is divided by b
    }
}

```

3.2.2.3 Pseudocode Schnorr Non-Interactive Zero-Knowledge Proof

Class **SchnorrNIZKP**:

```

Function generate_proof(a, v, g, p, q, t) {
    Compute public key A as the result of  $g^a \bmod p$ 
    Compute public key V as the result of  $g^v \bmod p$ 
    Generate challenge c using Fiat-Shamir transformation by
    concatenate the string of g, V, and A then hash it and perform modulo operation
    with  $2^t - 1$ 
    Compute the response r as the result of  $(v - a * c) \bmod q$ 
    Return the challenge c and the response r.
}

Function verify_proof(c, r, A, V, g, p) {

```

3.2.2.4 Pseudocode Penghitungan Suara Terenkripsi dan Dekripsi Total Suara

```

Function main() {
    Initialize person1_votes and person2_votes to 0
    Loop indefinitely {
        Prompt user to enter '1' to vote for Person 1, '2' to vote for
Person 2, or 'q' to quit
        If the entered vote is 'q', break the loop
        If the entered vote is '1' {
            Generate a random private key a and random v for
Person 1
            Generate a Schnorr NIZK proof for the vote
            If the proof is valid, increment person1_votes and
encrypt the vote using Paillier
            Display success or failure message based on NIZK
proof verification
        }
        If the entered vote is '2': {
            Generate a random private key a and random v for
Person 2

```


BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Kompleksitas Algoritma

Algoritma beroperasi dipengaruhi oleh berbagai faktor seperti spesifikasi mesin, alokasi memori pada satu waktu, dan faktor lainnya. Oleh karena itu, untuk mengevaluasi efisiensi suatu algoritma, lebih tepatnya menggunakan perhitungan kompleksitas waktu dengan mengukur jumlah operasi untuk setiap perintah pada algoritma tersebut. Pengukuran ini biasanya menggunakan notasi *big-theta* (θ).

4.1.1 Kompleksitas Algoritma pada Fungsi `random_odd_value`

Tabel 4.1 Tabel Kompleksitas Algoritma Fungsi `random_odd_value`

Function <code>random_odd_value(bits)</code> :	C	#	C#
Choose a random $n \in \mathbb{Z}, 2^{(\text{bits}-1)} \leq n < 2^{\text{bits}} - 1$	C1	1	C1
If n is even, increment it by 1	C2	1	C2
Return n	C3	1	C3

$$\begin{aligned} T(\text{bits}) &= C1 + C2 + C3 \\ &= \theta(1) \end{aligned}$$

4.1.2 Kompleksitas Algoritma pada Fungsi `is_prime`

Tabel 4.2 Tabel Kompleksitas Algoritma Fungsi `is_prime`

Function <code>is_prime(n, j=50)</code> :	C	#	C#
If n is less than or equal to 1, return False	C1	1	C1
If n is less than or equal to 3, return True	C1	1	C1

Initialize k to 0 and m to n - 1	C2	1	C2
While m is even:	C3	log n	C3(log n)
Increment k by 1	C4	log n	C4(log n)
Divide m by 2	C5	log n	C5(log n)
Repeat the following j times:	C6	1	C6
Choose a random a $\in \mathbb{Z}$, $2 \leq \mathbf{n} < \mathbf{n} - 2$	C7	1	C7
Calculate x as $\mathbf{a}^{\mathbf{m}} \bmod \mathbf{n}$	C8	1	C8
If x is equal to 1 or x is equal to n - 1, continue to the next iteration	C1	1	C1
For each i in the range 1 to k - 1:	C6	log n	C6(log n)
Square x and update its value ($\mathbf{x} = \mathbf{x}^2 \bmod \mathbf{n}$)	C9	log n	C9(log n)
If x is equal to n - 1, break the loop	C1	log n	C1(log n)
If x is not equal to n - 1, return False	C1	1	C1
Return True	C3	1	C3

$$\begin{aligned}
T(n, j) &= C1 + C1 + C2 + C3(\log n) + C4(\log n) + C5(\log n) + C6 + C7 \\
&\quad + C8 + C1 + C6(\log n) + C9(\log n) + C1(\log n) + C1 + C3 \\
&= (C1 + C3 + C4 + C5 + C6 + C9)(\log n) + 4C1 + C2 \\
&\quad + C3 + C6 + C7 + C8 \\
&= \theta(\log n)
\end{aligned}$$

4.1.3 Kompleksitas Algoritma pada Fungsi generate_prime

Tabel 4.3 Tabel Kompleksitas Algoritma Fungsi generate_prime

Function generate_prime(bits):	C	#	C#
Repeat indefinitely:	C1	1	C1
Generate a random odd value n from random_odd_value function with the specified number of bits	C2	1	C2
If n is a prime number based on the is_prime function, return n	C3	log n	C3(log n)

$$\begin{aligned}
 T(bits) &= C1 + C2 + C3(\log n) \\
 &= \theta(\log n)
 \end{aligned}$$

4.1.4 Kompleksitas Algoritma pada Fungsi generate_keys

Tabel 4.4 Tabel Kompleksitas Algoritma Fungsi generate_keys

Function generate_keys():	C	#	C#
Initialize bits to 128	C1	1	C1
Create a PrimeGenerator instance	C2	1	C2
Create a Paillier instance	C3	1	C3
Create a SchnorrNIZKP instance	C4	1	C4
Generate prime numbers p and q using the PrimeGenerator , but p is not equal to q	C5	log n	C5(log n)
Calculate n as p * q	C6	1	C6
Initialize g to 1 + n	C7	1	C7

Function generate_keys() :	C	#	C#
Initialize bits to 128	C1	1	C1
Create a PrimeGenerator instance	C2	1	C2
Create a Paillier instance	C3	1	C3
Create a SchnorrNIZKP instance	C4	1	C4
Generate prime numbers p and q using the PrimeGenerator , but p is not equal to q	C5	$\log n$	$C5(\log n)$
Calculate n as p * q	C6	1	C6
Initialize g to $1 + n$	C7	1	C7
Calculate lambda as $(p - 1) * (q - 1)$	C8	1	C8
Calculate k as the result of $(g^{\text{lambda}} \bmod n^2) - 1 / n$	C9	$\log n$	$C9(\log n)$
Calculate mu as the modular inverse of k mod n	C10	$\log n$	$C10(\log n)$

$$\begin{aligned}
 T() &= C1 + C2 + C3 + C4 + C5(\log n) + C6 + C7 + C8 + C9(\log n) \\
 &\quad + C10(\log n) \\
 &= \theta(\log n)
 \end{aligned}$$

4.1.5 Kompleksitas Algoritma pada Fungsi **find_gcd**

Tabel 4.5 Tabel Kompleksitas Algoritma Fungsi **find_gcd**

Function find_gcd(a, b) :	C	#	C#
While b is not zero:	C1	$\log(\min(a, b))$	$C1(\log(\min(a, b)))$

Set a to b	C2	$\log(\min(a, b))$	$C2(\log(\min(a, b)))$
Set b to the remainder when a is divided by b	C2	$\log(\min(a, b))$	$C2(\log(\min(a, b)))$

$$\begin{aligned}
 T(a, b) &= (C1 + C2 + C3)(\log(\min(a, b))) \\
 &= \theta(\log(\min(a, b)))
 \end{aligned}$$

4.1.6 Kompleksitas Algoritma pada Fungsi encrypt_message

Tabel 4.6 Tabel Kompleksitas Algoritma Fungsi encrypt_message

Function encrypt_message(m, r, g, n):	C	#	C#
Ensure that the greatest common divisor of r and n is 1	C1	$\log(\min(r, n))$	$C1(\log(\min(r, n)))$
Calculate c as the result of $g^m * r^n \pmod{n^2}$	C2	1	C2
Return c	C3	1	C3

$$\begin{aligned}
 T(m, r, g, n) &= C1(\log(\min(r, n))) + C2 + C3 \\
 &= \theta(\log(\min(r, n)))
 \end{aligned}$$

4.1.7 Kompleksitas Algoritma pada Fungsi decrypt_message

Tabel 4.7 Tabel Kompleksitas Algoritma Fungsi decrypt_message

Function decrypt_message(c, lambda, mu, n):	C	#	C#
Calculate u as the result of $(c^\lambda \pmod{n^2}) - 1 / n$	C1	$\log n$	$C1(\log n)$
Calculate p as the result of $u * \mu \pmod{n}$	C2	1	C2

Return p	C3	1	C3
-----------------	----	---	----

$$\begin{aligned}
 T(c, \lambda, \mu, n) &= C1(\log n) + C2 + C3 \\
 &= \theta(1)
 \end{aligned}$$

4.1.8 Kompleksitas Algoritma pada Fungsi generate_proof

Tabel 4.8 Tabel Kompleksitas Algoritma Fungsi generate_proof

Function generate_proof(a , v , g , p , q , t):	C	#	C#
Compute public key A as the result of g^a mod p	C1	log p	C1(log p)
Compute public key V as the result of g^v mod p	C2	log p	C2(log p)
Generate challenge c using Fiat-Shamir transformation by concatenate the string of g , V , and A then hash it and perform modulo operation with $2^t - 1$	C3	1	C3
Compute the response r as the result of (v - a * c) mod q	C4	1	C4
Return the challenge c and the response r .	C5	1	C5

$$\begin{aligned}
 T(a, v, g, p, q, t) &= C1(\log p) + C2(\log p) + C3 + C4 + C5 \\
 &= (C1 + C2)(\log p) + C3 + C4 + C5 \\
 &= \theta(\log p)
 \end{aligned}$$

4.1.9 Kompleksitas Algoritma pada Fungsi `verify_proof`

Tabel 4.9 Tabel Kompleksitas Algoritma Fungsi `verify_proof`

Function <code>verify_proof(c, r, A, V, g, p)</code> :	C	#	C#
Compute a verification value V by checking if $V = g^r * A^c \bmod p$	C1	1	C1
If the V holds, return True ; otherwise, return False .	C2	1	C2

$$\begin{aligned}
 T(c, r, A, V, g, p) &= C1 + C2 \\
 &= \theta(1)
 \end{aligned}$$

4.1.10 Kompleksitas Algoritma pada Fungsi Utama

Tabel 4.10 Tabel Kompleksitas Algoritma Fungsi utama

Function <code>main()</code> :	C	#	C#
Initialize person1_votes and person2_votes to 0	C1	1	C1
Generate keys for Paillier Cryptosystem and Schnorr NIZKP	C2	$\log n$	$C2(\log n)$
Loop indefinitely	C3	1	C3
Prompt user to enter '1' to vote for Person 1 , '2' to vote for Person 2 , or 'q' to quit	C4	1	C4
If the entered vote is 'q', break the loop	C5	1	C5
If the entered vote is '1'	C5	1	C5
Generate a random private key a and random v for Person 1	C6	1	C6

Generate a Schnorr NIZK proof for the vote	C7	log p	C7(log p)
If the proof is valid , increment person1_votes and encrypt the vote using Paillier	C5	log n	C5(log n)
Display success or failure message based on NIZK proof verification	C8	1	C8
If the entered vote is ' 2 '	C5	1	C5
Generate a random private key a and random v for Person 2	C9	1	C9
Generate a Schnorr NIZK proof for the vote	C10	log p	C10(log p)
If the proof is valid , increment person2_votes and encrypt the vote using Paillier	C5	log n	C5(log n)
Display success or failure message based on NIZK proof verification	C11	1	C11
Calculate the encrypted total votes as the product of encrypted_person1_votes and encrypted_person2_votes mod n^2	C12	1	C12
Decrypt the individual and total votes using Paillier	C13	1	C13

$$\begin{aligned}
T(n) &= C1 + C2(\log n) + C3 + C4 + C5 + C5 + C6 + C7(\log p) + C5(\log n) + C8 \\
&\quad + C5 + C9 + C10(\log p) + C5(\log n) + C11 + C12 + C13 \\
&= (C2 + 2C5)(\log n) + (C7 + C10)(\log p) + 3C5 + C1 + C3 + C4 + C6 + C8 + C9 \\
&\quad + C11 = C12 + C13 \\
&= \theta(\log n)
\end{aligned}$$

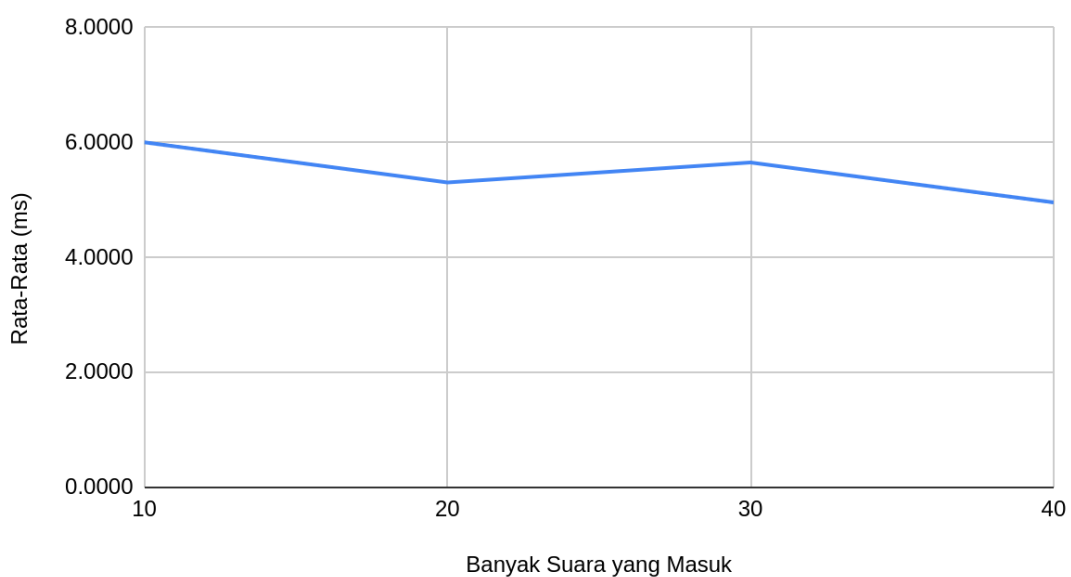
4.2 Pengujian *Real Running Time* pada Proses Perhitungan Suara Terenkripsi dan Dekripsi Total Suara

Pengujian *real running time* dilakukan pada proses perhitungan suara terenkripsi dan dekripsi total suara untuk menganalisa waktu proses berdasarkan banyaknya suara yang masuk dalam satuan waktu milidetik. Pengujian dilakukan dengan rentang 10 - 100 kali penghitungan suara.

Tabel 4.11 Tabel Pengujian *Real Running Time*

Banyak Suara yang Masuk	Percobaan 1 (ms)	Percobaan 2 (ms)	Percobaan 3 (ms)	Percobaan 4 (ms)	Percobaan 5 (ms)	Rata-Rata (ms)
10	6.3662	5.9311	5.1951	6.1883	6.3445	6.00504
20	5.2232	1.5892	6.8736	7.3058	5.5499	5.3083
30	5.2771	5.2452	5.9008	6.8750	4.9750	5.6546
40	5.3324	6.3054	5.5963	2.5143	5.0506	4.9598

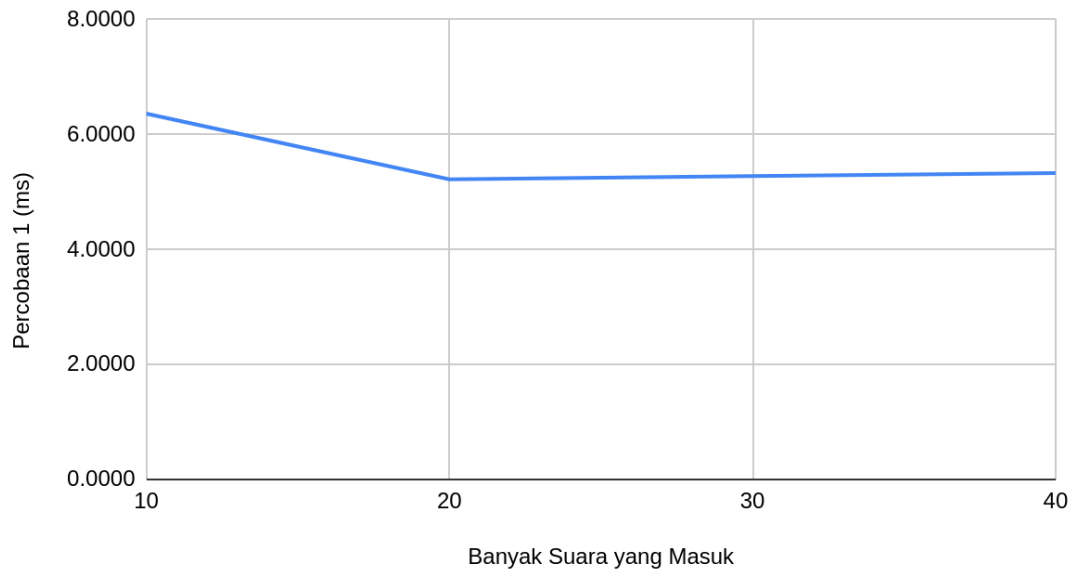
Rata-Rata (ms) vs Banyak Suara yang Masuk



Gambar 4.1 Grafik Pengujian Rata-Rata Waktu Dengan Banyak Suara yang Masuk

Pada gambar 4.1 memperlihatkan pengujian waktu pada proses perhitungan suara terenkripsi dan dekripsi total suara berdasarkan banyak suara yang masuk dan nilai rata-rata dari *running time*.

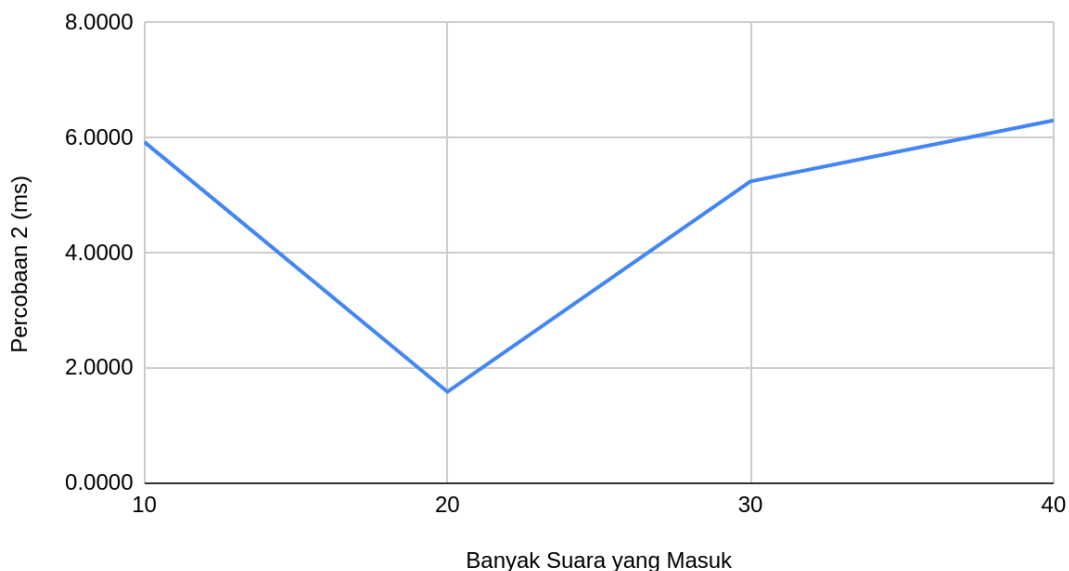
Percobaan 1 (ms) vs Banyak Suara yang Masuk



Gambar 4.2 Grafik pengujian waktu percobaan 1 dengan banyak suara yang masuk

Pada gambar 4.2 memperlihatkan pengujian waktu pada proses perhitungan suara ter enkripsi dan dekripsi total suara berdasarkan banyak suara yang masuk dan *running time* pada percobaan pertama.

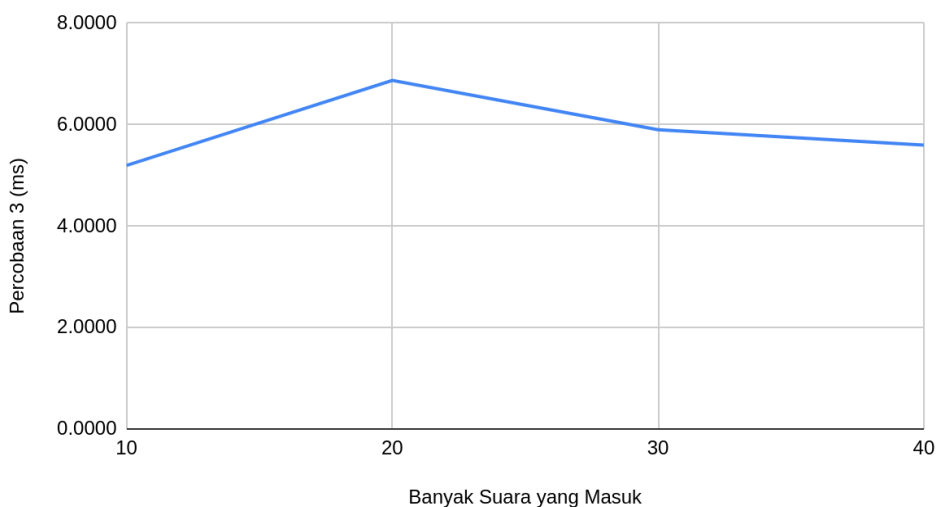
Percobaan 2 (ms) vs Banyak Suara yang Masuk



Gambar 4.3 Grafik pengujian waktu percobaan 2 dengan banyak suara yang masuk

Pada gambar 4.3 memperlihatkan pengujian waktu pada proses perhitungan suara ter enkripsi dan dekripsi total suara berdasarkan banyak suara yang masuk dan *running time* pada percobaan kedua.

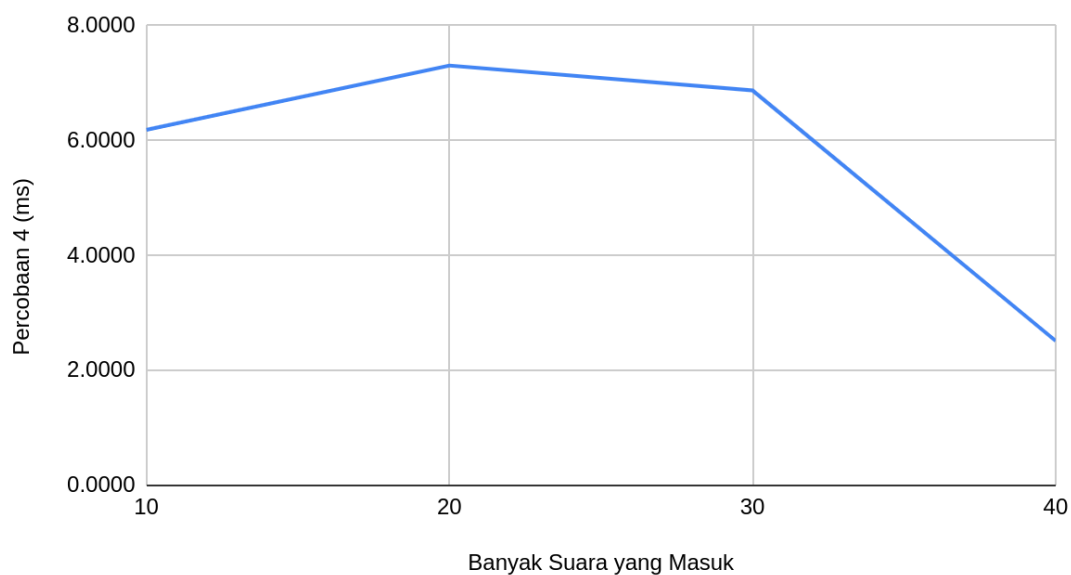
Percobaan 3 (ms) vs Banyak Suara yang Masuk



Gambar 4.4 Grafik pengujian waktu percobaan 3 dengan banyak suara yang masuk

Pada gambar 4.4 memperlihatkan pengujian waktu pada proses perhitungan suara ter enkripsi dan dekripsi total suara berdasarkan banyak suara yang masuk dan *running time* pada percobaan ketiga.

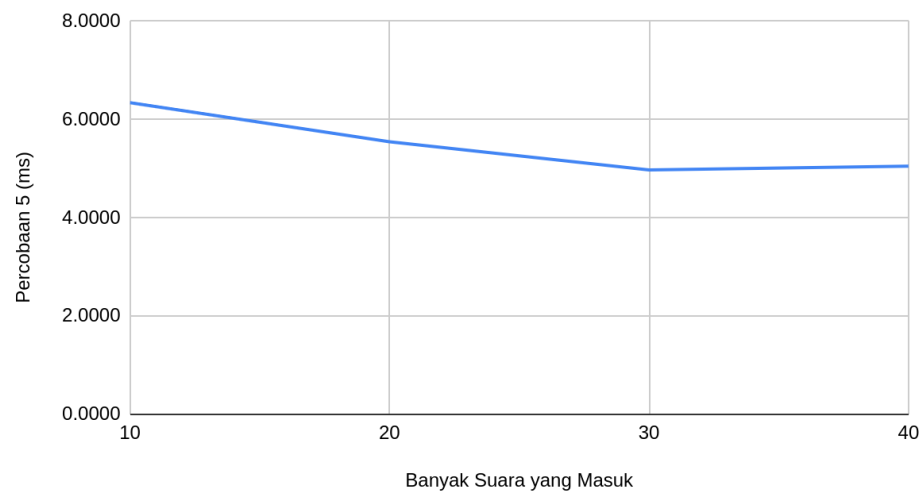
Percobaan 4 (ms) vs Banyak Suara yang Masuk



Gambar 4.5 Grafik pengujian waktu percobaan 4 dengan banyak suara yang masuk

Pada gambar 4.5 memperlihatkan pengujian waktu pada proses perhitungan suara terenkripsi dan dekripsi total suara berdasarkan banyak suara yang masuk dan *running time* pada percobaan keempat.

Percobaan 5 (ms) vs Banyak Suara yang Masuk

**Gambar 4.6** Grafik pengujian waktu percobaan 5 dengan banyak suara yang masuk

Pada gambar 4.6 memperlihatkan pengujian waktu pada proses perhitungan suara ter enkripsi dan dekripsi total suara berdasarkan banyak suara yang masuk dan *running time* pada percobaan kelima.

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat ditarik setelah melalui tahap analisis, perancangan, implementasi, dan pengujian *Paillier Cryptosystem* dan protokol *Schnorr Non-Interactive Zero-Knowledge Proof* dalam penghitungan dan validasi suara.

1. *Paillier Cryptosystem* dapat digunakan untuk penghitungan suara terenkripsi karena perkalian dua buah *ciphertext* akan menghasilkan penjumlahan dari *plaintext*-nya.
2. Protokol *Schnorr Non-Interactive Zero-Knowledge Proof* dapat digunakan untuk melakukan validasi suara dimana bukti (*proof*) dapat dihasilkan oleh sistem tanpa ada masukan dari pengguna.
3. *Paillier Cryptosystem* dan protokol *Schnorr Non-Interactive Zero-Knowledge Proof* memiliki kompleksitas waktu $\theta(\log n)$.
4. *Paillier Cryptosystem* dan protokol *Schnorr Non-Interactive Zero-Knowledge Proof* memiliki *running time* rata-rata 5,4819 ms.

5.2 Saran

Berikut adalah beberapa saran yang dapat dipertimbangkan untuk penelitian lebih lanjut.

1. Penambahan fitur autentikasi pengguna untuk mensimulasikan *e-voting* dengan lebih baik.
2. Menggunakan *Fully Homomorphic Encryption* dengan mempertimbangkan kecepatan enkripsi dan penghitungan suara terenkripsi.

DAFTAR PUSTAKA

- Yi, X., Paulet, R., & Bertino, E. (2014). Homomorphic Encryption and Applications. Springer.
- Goldreich, O. (2001). Foundations of Cryptography: Volume 1, Basic Tools. https://openlibrary.org/books/OL7754136M/Foundations_of_Cryptography
- Fiat, A., & Shamir, A. (n.d.). How To Prove Yourself: Practical Solutions to Identification and Signature Problems. Lecture Notes in Computer Science, 186–194. doi:10.1007/3-540-47721-7_12
- Sapuroo, S., Warke, V., Pote, S., & Dhumal, R. (2020). Online Voting System using Homomorphic Encryption. ITM Web of Conferences, 32, 03023. <https://doi.org/10.1051/itmconf/20203203023>
- Raut, B., Jagtap, M., Ghule, S., Jadhav, K., Amdhakar, S. P. (2020). Secure E - Voting System Based on Paillier Cryptography. International Journal of Scientific Research in Computer Science, Engineering and Information Technology.
- Hao, F. (2017). Schnorr non-interactive zero-knowledge proof (No. rfc8235).
- Alharbi, A., Zamzami, H., & Samkri, E. (2020). Survey on Homomorphic Encryption and Address of New Trend. International Journal of Advanced Computer Science and Applications, 11.
- Katz, J., & Lindell, Y. (2014). *Introduction to modern Cryptography*. CRC Press.
- Aumasson, J. (2017). *Serious cryptography: A Practical Introduction to Modern Encryption*. No Starch Press.
- Stallings, W. (2022). *Cryptography and Network Security: Principles and Practice*. Prentice Hall.
- Gentry, Craig (2009). A Fully Homomorphic Encryption Scheme. Stanford University.
- Shinde Shubhangi dkk (2013). Secure E-voting Using Homomorphic Technology. Terna Engineering College.
- Paillier, Pascal (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Gemplus Card International.

LAMPIRAN OUTPUT PROGRAM

Login

Email

Password

Login

Don't have an account? [Create a new account](#)

Welcome, bencbatch!

Email: bencbatch@gmail.com

User Id: 1

Logout

Vote

Get Votes

Create Ballot

Vote

Ballot Id

Creator Id

Vote Choice

Submit Vote

Back to homepage

Get votes

Get Votes

Back to homepage

Create Ballot

Email

Add Another Email

Create Ballot

Back to homepage

DATABASE

ElephantSQL secure-vote Rekyu

DETAILS
ALARMS
BROWSER
STATS
SLOW QUERIES
BACKUP
LOG
METRICS
ADMIN
INTEGRATIONS
FIREWALL
MAINTENANCE

SQL Browser cuthmdkv

SELECT * FROM "public"."ballot" LIMIT 100

Table queries Previous queries Execute

id	creator_id	ballot_id	user_id	voter_id	encrypted_vote_1	encrypted_vote_2	encrypted_total_votes
1	1	1	1		337767000476002255089323280141053422225989132 832003578325724056252425541413678328888962771 5340783954305820168374151261919888784035626610 111999732010079916	112254700477077058030924420683025451963958385 582762879362704740673815231134180829943559834 526481469381325576802207678989573561204879775 2083276414286500353	175406060381595777008214614104076196 52541672185142270086197738962045489341 050458571924236202777685496308900531 99379695221745706
2	1	1	2		259250498531994632328224534877094590571358753 59044185761835166215992550454065557641369253 537529704654702475963778908841287621008507005 71642971369890047796	210950846597344042108072433546078333843438124 1337899134423273558412522747782269247919473498 63206719072072506389726134238366546498315461 922916380769641631	375703681035574808187262034030364487 528009515475282809825632093547657977 261666363552586835136044865076699225 06764055865064775
3	1	1	3		154567663432207718212855893962130138008513670 6133789335133656178232497969449276903972027071 120506897058084373577323011038869105565079471 4945706554403467	289806967103340304117429780108564006824379478 569760612068690241752203520698941677539559014 083778244327829430529981731493487199622249672 6806445936588929272	101536397998522539370887731806054873 143326502780632992324874720196729020 682635418024417509674758916899090502 262693785803692614