TUGAS AKHIR – IF184802

PENGGUNAAN *RANDOM ORTHOGONAL MATRIX MASKING* UNTUK *MELINDUNGI DATA CONFIDENTIALITY* PADA STUDI KASUS *E-VOTING*

VINSENSIUS INDRA SURYANTO
NRP 05111640000064

Dosen Pembimbing I
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II
Ir. F.X. Arunanto, M.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

TUGAS AKHIR – IF184802

PENGGUNAAN *RANDOM ORTHOGONAL MATRIX MASKING* UNTUK MELINDUNGI *DATA CONFIDENTIALITY* PADA STUDI KASUS *E-VOTING*

VINSENSIUS INDRA SURYANTO
NRP 05111640000064

Dosen Pembimbing I
Rully Soelaiman, S.Kom., M.Kom.

Dosen Pembimbing II
Ir. F.X. Arunanto, M.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

UNDERGRADUATE THESIS – IF184802

RANDOM ORTHOGONAL MATRIX MASKING IMPLEMENTATION FOR PROTECTING DATA CONFIDENTIALITY ON E-VOTING CASE STUDY

VINSENSIUS INDRA SURYANTO
NRP 05111640000064

First Advisor
Rully Soelaiman, S.Kom., M.Kom.

Second Advisor
Ir. F.X. Arunanto, M.Sc.

Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

PENGGUNAAN *RANDOM ORTHOGONAL MATRIX MASKING* UNTUK MELINDUNGI *DATA CONFIDENTIALITY* PADA STUDI KASUS *E-VOTING*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Algoritma dan Pemrograman
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
VINSENSIUS INDRA SURYANTO
NRP: 05111640000064

Disetujui oleh Pembimbing Tugas Akhir:

1. Rully Soelaiman, S.Kom., M.Kom. .....
   (NIP. 197002131994021001) (Pembimbing 1)

2. Ir. F.X. Arunanto, M.Sc. .....
   (NIP. 195701011983031004) (Pembimbing 2)

SURABAYA
Juni, 2020

[*Halaman ini sengaja dikosongkan*]

PENGGUNAAN *RANDOM ORTHOGONAL MATRIX MASKING* UNTUK MELINDUNGI *DATA CONFIDENTIALITY* PADA STUDI KASUS *E-VOTING*

Nama : Vinsensius Indra Suryanto
NRP : 05111640000064
Departemen : Teknik Informatika, FTEIC-ITS
Pembimbing I : Rully Soelaiman, S.Kom., M.Kom.
Pembimbing II : Ir. F.X. Arunanto, M.Sc.

ABSTRAK

Dalam hal pemungutan suara elektronik, penyelenggara harus memiliki protokol yang aman serta dapat menjamin bahwa setiap surat suara harus sampai ke tangan yang seharusnya. Hal ini disebabkan karena pertukaran data pada sistem pemungutan suara elektronik mengandung hal yang sensitif. Banyak model protokol pemungutan suara elektronik yang ada namun tidak semua protokol tersebut dapat menjamin prinsip-prinsip dalam pelaksanaan pemungutan suara elektronik.

Pada tugas akhir ini, penulis melakukan implementasi dan simulasi terhadap sistem pemungutan suara elektronik yang mengacu pada protokol MVP. Model Voting Protocol (MVP) merupakan protokol pemungutan suara daring terbaru yang menggunakan random orthogonal matrix masking (ROMM) sebagai metode dalam pengumpulan data.

Protokol MVP diimplementasikan dengan menggunakan bahasa pemrograman Python dan R. Setelah melakukan implementasi dan simulasi, didapatkan kesimpulan bahwa protokol MVP dapat menjamin anonimitas dan prinsip-prinsip dalam pemungutan suara elektronik, serta biaya komputasi yang cukup rendah.

Kata Kunci: ***Electronic Voting, Model Voting Protocol, Statistical Disclosure Limitation***.

RANDOM ORTHOGONAL MATRIX MASKING IMPLEMENTATION FOR PROTECTING DATA CONFIDENTIALITY ON E-VOTING CASE STUDY

Student's Name : Vinsensius Indra Suryanto
Student's ID : 05111640000064
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Rully Soelaiman, S.Kom., M.Kom.
Second Advisor : Ir. F.X. Arunanto, M.Sc.

ABSTRACT

In the case of electronic voting, the organizer must have a secure protocol and can guarantee that each ballot must get to the hands that it should. This is because the exchange of data on electronic voting systems contains sensitive matters. Many electronic voting protocol models exist, but not all these protocols can guarantee the principles of implementing online voting.

In this final project, the author implements and simulates an electronic voting system that refers to the MVP protocol. Model Voting Protocol (MVP) is the latest online voting protocol that uses random orthogonal matrix masking (ROMM) as a method of collecting data.

The MVP protocol is implemented using the Python and R. programming languages. By conducting the implementation and simulation, it is concluded that the MVP protocol can guarantee anonymity and principles in online voting, and the cost of computing is quite low.

Keywords: ***Electronic Voting, Model Voting Protocol, Statistical Disclosure Limitation***.

KATA PENGANTAR

Puji syukur saya sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya, penulis dapat melaksanakan tugas akhir yang berjudul:

"**PENGGUNAAN *RANDOM ORTHOGONAL MATRIX MASKING* UNTUK MELINDUNGI *DATA CONFIDENTIALITY* PADA STUDI KASUS *E-VOTING***"

Pembuatan tugas akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Orang tua dan keluarga penulis, yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Bapak Rully Soelaiman, S.Kom., M.Kom., selaku pembimbing I yang telah memberikan arahan, motivasi, nasihat dan bimbingan dalam menyelesaikan tugas akhir ini.
3. Bapak Ir. F.X. Arunanto, M.Sc., selaku pembimbing II yang telah membimbing dan memberikan dukungan dalam menyelesaikan tugas akhir ini.
4. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua Departemen Teknik Informatika ITS dan seluruh dosen dan karyawan Departemen Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Departemen Teknik Informatika ITS.
5. Teman-teman mahasiswa Teknik Informatika ITS khususnya mereka yang dibawah bimbingan Bapak Rully Soelaiman, yang bersama-sama telah menjadi teman dan

pendukung penulis selama masa perkuliahan dan pengerjaan tugas akhir.

6. Seluruh mahasiswa Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Informatika ITS.
7. Serta seluruh pihak yang telah turut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa laporan tugas akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis ke depannya. Selain itu, penulis berharap laporan tugas akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Juni 2020

DAFTAR ISI

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Internet membuat pertukaran informasi menjadi sangat luas dan bisa didapatkan di mana saja. Namun perlu diperhatikan bahwa pertukaran informasi tersebut dapat memuat data yang memiliki kerahasiaan. Berbagai macam pendekatan dapat dilakukan untuk meningkatkan proteksi data. Beberapa di antaranya adalah dengan menggunakan kriptografi, *statistical disclosure limitation* (SDL), *privacy-preserving data mining* (PPDM), maupun membatasi hak akses data. Pada tugas akhir ini, penulis mengangkat topik pada *statistical disclosure limitation* yang berupa *matrix masking*. Matrix *masking* merupakan salah satu metode dalam *statistical disclosure limitation* yang berfungsi untuk mempertahankan kerahasiaan data, serta mencegah adanya kebocoran data dalam pendistribusiannya.

Pengembangan internet menghasilkan beberapa platform pemungutan suara elektronik yang memudahkan rangkaian proses pemungutan suara, sehingga peserta pemungutan suara tidak perlu datang secara langsung ke tempat pemungutan suara, namun cukup dengan menggunakan perangkat ponsel atau komputer, proses pemungutan suara dapat dilaksanakan.

Pemungutan suara elektronik pada umunya mengadaptasi perilaku pemungutan suara tradisional (*paper based*), sehingga perlu diperhatikan pula sifat-sifat khusus dan anonimitas pada saat merancang proses pemungutan suara elektronik. Oleh karena itu, untuk membangun sebuah pemungutan suara elektronik yang anonim, dibutuhkan protokol yang dapat menjamin kebenaran dan kerahasiaan data pada setiap prosesnya.

Salah satu studi kasus dari pemungutan suara elektronik dalam dunia nyata adalah adanya *Electronic Voting Machine* (EVM). EVM merupakan mesin khusus pemungutan suara elektronik yang sekarang sudah dipakai untuk melakukan

pemungutan suara di India [1]. EVM memungkinkan terciptanya lingkungan pemungutan suara yang aman dan terkendali, serta memberikan kemudahan penggunaan bagi pesertanya. Namun EVM juga memiliki beberapa kekurangan yang perlu diperhatikan. Karena EVM merupakan alat/perangkat keras yang memberi fasilitas pemungutan suara, maka alat tersebut memiliki tantangan yang berupa ketahanan alat tersebut dari faktor alamiah seperti bencana dan umur alat. Dalam pelaksanaanya, EVM juga menuai beberapa kritik seperti kurangnya transparansi pada protokolnya, serta kurangnya fleksibilitas pada pelaksanaanya.

Studi literatur juga dilakukan untuk mencari berbagai sumber lain mengenai protokol pemungutan suara elektronik yang sudah pernah ada. Protokol-protokol lain yang ditemukan dapat dikelompokkan berdasarkan penggunaan metodenya: *mix-net*, *blind signature*, dan enkripsi *homomorphic* [2]. Protokol pemungutan suara elektronik berbasis *mix-net* dapat mencapai anonimitas dengan memanfaatkan saluran komunikasinya. Kekurangan dari penggunaan *mix-net* adalah beban komputasinya yang cukup mahal untuk membuktikan kebenaran dan ketergantungan pada banyak *mixer*, serta kebutuhan *deployment* pada *server*. Sedangkan untuk *blind signature* dan enkripsi *homomorphic* adalah kurangnya anonimitas pada saluran komunikasi dan kurang terjaminnya integritas pada *proxy server*.

Pada tugas akhir ini, penulis mengangkat topik berupa protokol pemungutan suara elektronik yang menggunakan teknik *statistical disclosure limitation* berupa *matrix masking*. *Matrix masking* merupakan salah satu metode yang berfungsi untuk mempertahankan kerahasiaan data serta mencegah adanya kebocoran data dalam pendistribusiannya. Metode tersebut memberikan *statistical disclosure limitation* pada data mikro dengan mempertahankan representasi konten aslinya.

Hasil yang diharapkan dari tugas akhir ini adalah menghasilkan platform pemungutan suara elektronik yang dapat menjamin kebenaran dan kerahasiaan dalam pendistribusian data, serta memiliki biaya komputasi yang cukup rendah.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana implementasi untuk mendapatkan suatu platform pemungutan suara elektronik yang efisien, aman, dan dapat menjamin kerahasiaan?
2. Bagaimana metode *matrix masking* dapat diterapkan sebagai penambah *statistical disclosure limitation* pada data pemungutan suara elektronik?
3. Bagaimana kinerja sistem yang telah dibangun pada saat proses pengujian?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. *Dataset* yang dipakai bersumber dari situs *kaggle*, 2016 *US Election*.
2. Analisis keamanan dilakukan pada protokol pemungutan suara yang diimplementasikan, dengan memberlakukan asumsi protokol tersebut dijalankan di saluran jaringan yang aman dan dapat dipercaya.
3. Menggunakan Python 3 sebagai bahasa pemrograman utama dan R sebagai bahasa pemrograman tambahan dalam implementasinya.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Memberikan kontribusi dengan menciptakan platform pemungutan suara daring yang aman, efisien, serta dapat menjamin kerahasiaan data.
2. Menerapkan *statistical disclosure limitation* berupa *matrix masking* pada data pemungutan suara elektronik guna meningkatkan kerahasiaan data.

3. Menciptakan platform pemungutan suara yang memiliki biaya komputasi yang relatif rendah.

1.5 Manfaat

Dengan adanya tugas akhir ini, diharapkan dapat membantu proses pengembangan terciptanya platform pemungutan suara elektronik yang efisien, aman dan anonim, serta memiliki biaya komputasi yang cukup rendah.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari tugas akhir ini adalah penyusunan Proposal tugas akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam tugas akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya tugas akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan tugas akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Terdapat pula subbab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan tugas akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur daring tambahan terkait.

1.6.3 Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem menggunakan Python 3 sebagai bahasa pemrograman utama, dan

R sebagai bahasa pemrograman pendukung, beserta *library* pendukung lainnya.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan dengan menggunakan data sintetis yang dibuat dari *dataset*. Evaluasi dilakukan dengan melakukan simulasi terhadap sistem yang dibuat, dan menguji sistem dengan menerapkan beberapa skenario.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan tugas akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.

Bab II Dasar Teori

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan tugas akhir ini. Secara garis besar, bab ini berisi tentang dasar teori kebutuhan dan protokol pemungutan suara elektronik serta *library* yang digunakan.

Bab III Desain

Bab ini berisi pembahasan mengenai perancangan dari protokol pemungutan suara elektronik dengan mengacu pada protokol MVP beserta perancangan pengolahan terhadap *dataset* yang akan diujikan.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode sumber yang digunakan untuk implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan tugas akhir, dan saran untuk pengembangan solusi ke depannya.

BAB II
DASAR TEORI

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam tugas akhir. Teori-teori tersebut berupa protokol dan kebutuhan dalam pemungutan suara elektronik, enkripsi dalam kriptografi, dan beberapa teori lain yang mendukung pembuatan tugas akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

2.1 Pemungutan Suara Elektronik (*E-voting*)

Pemungutan suara elektronik (*E-voting*) merupakan proses pemungutan dan penghitungan suara oleh suatu penyelenggara atau instansi dengan menggunakan perantara perangkat elektronik.

Secara garis besar, proses pelaksanaan pemungutan suara elektronik dibagi menjadi tiga tahapan [3]:

1. *Pre-voting*
   Pada tahap ini, terdapat proses registrasi peserta beserta kandidat yang akan dipilih, pertukaran kunci antar entitas, dan penyetelan sistem pemungutan suara.
2. *Voting*
   Pada tahap ini, peserta yang terdaftar sebagai pemilih dapat mengirimkan pilihannya melalui surat suara yang diproduksi oleh penyelenggara pemungutan suara. Pilihan yang diterima oleh penyelenggara dilakukan verifikasi terlebih dahulu, dan kemudian dikumpulkan dalam kotak suara digital.
3. *Post-voting*
   Tahap ini merupakan tahap yang dilakukan setelah waktu pemungutan suara telah usai. Pada tahap ini, dilakukan penghitungan suara oleh penyelenggara.

2.2 Kebutuhan pada Penyedia Layanan Pemungutan Suara Elektronik

Penyelenggara pemungutan suara elektronik membutuhkan suatu protokol yang mengandung beberapa sifat khusus agar dapat menjamin keberlangsungan, kebenaran dan kerahasiaan suara dalam proses pemungutan suara. Selain dijalankan pada komunikasi dan komponen yang aman, sebuah protokol pemungutan suara elektronik yang kriptografis yang aman harus memenuhi beberapa persyaratan sebagai berikut [4]:

1. *Voter Privacy*
   Merupakan sebuah pencegahan untuk mengasosiasikan seorang pemilih dengan suara yang diberikannya.
2. *Eligibility*
   Hanya pemilih terdaftar yang dapat memberikan suara pada saat proses pemungutan suara berlangsung. Pemilih terdaftar ialah mereka yang telah melakukan registrasi kepada pihak penyelenggara, pada saat sebelum pemungutan suara berlangsung.
3. *Uniqueness*
   Dalam satu pemungutan suara, satu pemilih harus memberikan hanya satu suara.
4. *Fairness*
   Tidak ada perhitungan parsial yang terungkap pada saat proses pemungutan suara belum berakhir. Hal ini dilakukan untuk memberikan keputusan yang adil bagi semua kandidat.
5. *Uncoercibility*
   Segala bentuk usaha paksaan dari pihak manapun, tidak dapat memengaruhi nilai suara yang diberikan. Setiap pemilih harus memberikan suaranya secara bebas.
6. *Receipt-freeness*
   Merupakan suatu indikasi bahwa sistem tidak memberikan konfirmasi penerimaan suara yang dapat menghasilkan kontennya. Dengan kata lain, pemilih tidak bisa membuktikan bahwa yang bersangkutan telah memilih suatu

kandidat secara spesifik, baik selama proses pemungutan suara maupun setelahnya. Hal ini dilakukan sebagai pencegahan terjadinya jual beli suara.

7. *Accuracy*
   Hasil penghitungan akhir pemungutan suara, merupakan hasil yang benar dan akurat sesuai dengan banyaknya suara yang terkumpul.

2.3 *Model Voting Protocol* (MVP)

Model Voting Protocol (MVP) merupakan sebuah protokol pemungutan suara elektronik yang efisien dan anonim yang memanfaatkan teknik *dual random matrix masking* (DRMM) sebagai metode untuk mengemas suara pada protokolnya [5]. DRMM menggunakan *random orthogonal matrix masking* (ROMM) sebagai *statistical disclosure limitation*. Dalam arsitektur MVP, terdapat 3 entitas, di antaranya:

1. *Voter*
   Merupakan peserta pemungutan suara yang telah terdaftar. Setiap peserta hanya diperbolehkan mengirimkan satu suara yang valid.
2. *Data Management Center* (DMC)
   Sebuah entitas atau organisasi yang menyelenggarakan dan mengatur proses pemungutan suara. DMC merupakan pusat dari pelaksanaan pemungutan suara, dan bertanggung jawab atas autentikasi peserta, penghitungan suara, dan pengumuman atas hasil pemungutan suara.
3. *Voting Proxy Server* (VPS)
   Merupakan sebuah server atau entitas yang bertanggung jawab sebagai tempat pengumpulan suara dari peserta yang telah terdaftar. Peserta mengirimkan suara yang terenkripsi ke VPS yang kemudian divalidasi terlebih dahulu. Pada saat proses pemungutan suara berakhir, VPS meneruskan *masked group* yang merupakan kumpulan suara kepada DMC.

Pelaksanaan rangkaian proses pemungutan suara dalam protokol MVP dibagi menjadi 4 fase:

1. Fase Inisialisasi
   Fase ini merupakan tahap awal dari pelaksanaan pemungutan suara elektronik. Setiap entitas melakukan pembuatan pasangan *asymmetric keys*, dan sebuah *symmetric key*. Pada fase ini juga dilakukan proses registrasi peserta, dan setiap peserta yang melakukan registrasi akan mendapat informasi registrasi yang nantinya digunakan pada proses autentikasi.
2. Fase Autentikasi
   Untuk menjamin hanya pemilih terverifikasi yang diizinkan memberikan suara, pemilih harus melakukan autentikasi diri mereka dengan DMC untuk mendapatkan informasi yang diperlukan untuk pemungutan suara. Dalam fase autentikasi, setiap pemilih akan melakukan autentikasi dengan memberikan informasi registrasi yang didapat sewaktu melakukan registrasi pada fase inisialisasi. Setelah peserta melakukan autentikasi, DMC melakukan pembuatan *ballot* (surat suara) untuk peserta yang bersangkutan. *Ballot* tersebut merupakan *ballot* yang telah diberi *noise* dan telah dilakukan proses *masking* dan enkripsi.
3. Fase *Voting*
   Fase ini memungkinkan peserta pemungutan suara memilih kandidat dari *ballot* yang telah diterima yang kemudian dipilih akan dikirimkan ke VPS. VPS mengumpulkan surat suara terenkripsi tersebut dan mengelompokkannya berdasarkan grup pemilih apabila la identitas yang diterima dari peserta adalah valid dan peserta tersebut belum melakukan pemilihan. Ketika proses pemungutan suara telah berakhir, VPS akan melakukan pembuatan *group ballot*. *Group ballot* tersebut dibentuk dengan melakukan agregasi pada setiap grupnya yang merupakan kumpulan dari suara yang sah. Kemudian VPS melakukan *matrix masking* pada setiap *group ballot* yang terbentuk dari fase

sebelumnya, yang kemudian diserahkan kepada DMC untuk dilakukan rekapitulasi hasil pemungutan suara.

4. Fase Penghitungan Suara
   Pada fase ini, dengan data yang dikirim oleh VPS, DMC melakukan perkalian matriks invers di setiap grupnya dan melakukan pengurangan dengan agregat *noise* awal. Dengan melakukan langkah itu DMC dapat mengetahui hasil dari pemungutan suara.

2.4 *Matrix Masking*

Matrix Masking merupakan salah satu metode *Statistical Disclosure Limitation* (*SDL*) yang berfungsi untuk melindungi kerahasiaan data statistik, melalui beberapa matriks spesifik mengubah matriks data menjadi matriks *mask* melalui pra dan pasca multiplikasi serta kemungkinan adanya penambahan *noise* atau *perturbation*. Sebagai contoh, Duncan [6] mengusulkan untuk mengubah sebuah $n \times p$ (kasus dengan variabel) data matriks awal X ke *masked data* dalam bentuk:

$$X^* \rightarrow AXB + C$$

dengan matriks A adalah operator baris, matriks B adalah operator kolom, dan matriks C adalah derau atau gangguan yang ditambahkan ke data.

2.4.1 *Random Orthogonal Matrix Masking*

Matriks ortogonal adalah matriks persegi yang kolom dan barisnya merupakan vektor satuan ortogonal, yaitu:

$$Q^T Q = Q Q^T = I$$

di mana *I* adalah matriks identitas.

Ini mengarah ke karakterisasi dimana matriks Q adalah ortogonal jika memiliki *transpose* sama dengan kebalikannya:

$$Q^T = Q^{-1}$$

Matriks ortogonal Q memiliki syarat harus dapat dibalik. Determinan matriks ortogonal adalah +1 atau −1. Sebagai transformasi linear, matriks ortogonal mempertahankan produk titik vektor, dan karena itu bertindak sebagai isometri ruang Euclidean, seperti rotasi, refleksi atau rotasi.

Ting et al. mengusulkan metode perturbasi baru yang disebut *Random Orthogonal Matrix Masking* (ROMM) untuk memberikan privasi kepada data mikro. ROMM adalah kasus khusus dari *matrix masking* yang standar, dengan syarat bahwa *B* adalah matriks identitas, *C* adalah matriks nol, dan *A* adalah beberapa matriks ortogonal acak yang diambil dari beberapa distribusi *D*. Oleh karena itu, transformasi ROMM adalah:

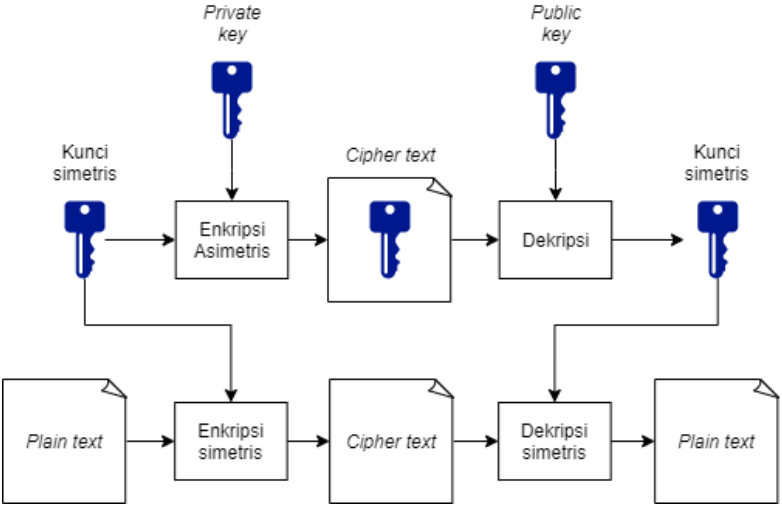
$$X \rightarrow AX$$

ROMM memiliki sebuah karakteristik yang penting, yaitu mempertahankan *sample mean* dan *sample covariance* setelah *masking* pra-matriks ortogonal.

2.5 Enkripsi Hibrid

Enkripsi hibrid adalah teknik enkripsi yang menggabungkan dua atau lebih sistem enkripsi [7]. Enkripsi hibrid menggabungkan kombinasi enkripsi asimetris dan simetris untuk mendapatkan manfaat dari kekuatan setiap bentuk enkripsi. Kekuatan ini masing-masing didefinisikan sebagai kecepatan dan keamanan.

Enkripsi hibrida dianggap sebagai jenis enkripsi yang sangat aman selama kunci publik dan pribadi sepenuhnya aman. Kelebihan lain dari enkripsi hibrid adalah menutupi kekurangan pada enkripsi asimetris saat enkapsulasi data, dimana pada enkripsi asimetris, jumlah panjang bit data harus tidak melebihi jumlah panjang bit kunci.



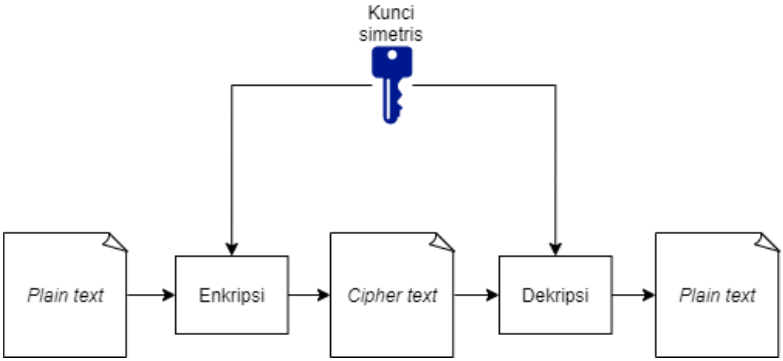
Gambar 2.1 Skema enkripsi hibrid

Masalah tersebut teratasi dengan menggunakan enkripsi simetris pada enkapsulasi data. Adapun gambaran umum proses yang terjadi pada skema kriptosistem hibrid dapat dilihat pada Gambar 2.1. Sebuah kriptosistem hibrid dapat dibangun menggunakan dua kiptosistem yang terpisah:

1. Skema enkapsulasi kunci, yang merupakan bagian dari kriptosistem asimetris, dan
2. Skema enkapsulasi data, yang merupakan bagian dari kriptosistem simetris.

2.6 Enkripsi Simetris

Enkripsi simetris umumnya menggunakan algoritma di mana hanya terdapat sebuah kunci (*secret key*) yang digunakan untuk mengenkripsi dan mendekripsi informasi elektronik. Entitas yang berkomunikasi melalui enkripsi simetris harus bertukar kunci sehingga dapat digunakan dalam proses dekripsi. Enkripsi simetris memiliki 5 komponen utama yang dapat dilihat pada Gambar 2.2 [8].



Gambar 2.2 Skema enkripsi simetris

Plaintext merupakan pesan awal yang dapat dibaca dan sebagai masukan algoritma yang akan dilakukan. *Encryption Algorithm* adalah algoritma yang melakukan proses substitusi dan transformasi pada pesan awal. *Ciphertext* merupakan pesan yang dibentuk dari pesan awal dan sebuah kunci rahasia, sehingga wujudnya bersifat acak dan tidak dapat diinterpretasikan. *Decryption algorithm* pada dasarnya adalah algoritma enkripsi yang dijalankan secara terbalik. Algoritma dekripsi berfungsi mengembalikan *ciphertext* kembali ke bentuk pesan awal. Terdapat beberapa algoritma yang termasuk dalam enkripsi simetris; Blowfish, AES, RC4, DES, RC5, dan RC6.

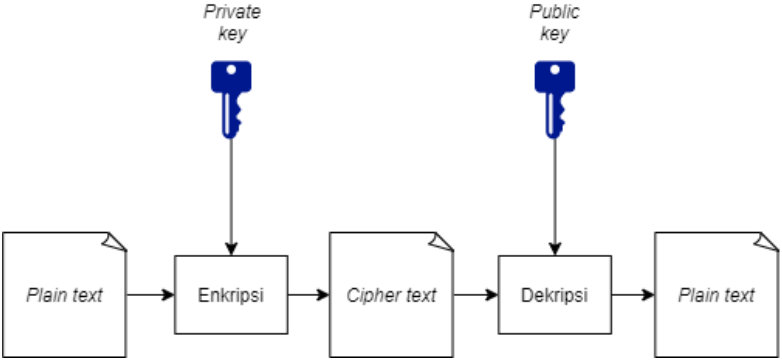
2.6.1 *Advanced Encryption Standard*

Advanced Encryption Standard (AES) adalah salah satu algoritma enkripsi simetris yang telah diterapkan menjadi standar enkripsi dalam dunia kriptografi. Algoritma ini diciptakan oleh Joan Daemen dan Vincent Rijmen. AES dapat memproses blok data sebanyak 128-bit dengan menggunakan panjang kunci 128, 192, atau 256-bit. Untuk panjang kunci 128, 192 dan 256 bit, masing-masing dapat disebut sebagai AES-128, AES-192 dan AES-256. Tidak seperti DES, AES tidak memiliki struktur *fiestel*. Jumlah putaran di AES tergantung pada panjang kunci. Misal untuk panjang kunci 128, jumlah putaran adalah 10 dan sama untuk

192 dan 256-bit, adalah 12 dan 14. AES memberikan pertahanan terhadap semua serangan yang telah diketahui, desainnya sederhana, dan memiliki kecepatan komputasi yang baik.

2.7 Enkripsi Asimetris

Enkripsi asimetris atau dikenal dengan kriptosistem *public key*, merupakan serangkaian proses yang menggunakan sepasang *key* yang berkaitan, sebuah *public key* dan sebuah *private key*, untuk melakukan enkripsi dan dekripsi pada suatu pesan dan bertujuan untuk memberi kerahasiaan dan batasan akses pada pesan tersebut. *Private key* merupakan kunci rahasia yang hanya dimiliki oleh penerbit awal pembuat pasangan kunci. Sedangkan *public key* merupakan kunci yang disebarkan ke pasangan entitas pembuat kunci awal. Untuk mendapatkan manfaat dari enkripsi asimetris pada saat berkomunikasi, setiap entitas menggunakan *key* yang mereka punya. Skema dari enkripsi asimetris dapat dilihat pada Gambar 2.3.



Gambar 2.3 Skema enkripsi asimetris

Beberapa algoritma seperti Diffie-Hellman, El-Gamal, dan RSA, merupakan implementasi dari enkripsi asimetris.

2.7.1 Rivest-Shamir-Adleman

Rivest-Shamir-Adleman (RSA) adalah salah satu algoritma dari beberapa enkripsi asimetris yang populer. Algoritma ini berdasar pada masalah *prime factorization*, yaitu mencari sepasang faktor prima pada suatu bilangan komposit yang sangat besar. Pada implementasinya, algoritma tersebut akan menghasilkan bilangan komposit yang sangat besar, dengan faktor prima dari bilangan tersebut merupakan komponen rahasia yang berperan penting dalam proses enkripsi dan dekripsi.

2.8 Python

Python adalah bahasa pemograman yang populer. Python sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa Inggris [9].

2.9 R

R merupakan sebuah bahasa pemrograman yang sekaligus menyediakan lingkungan pemgembangan perangkat lunak secara gratis untuk komputasi statistika dan grafik. R adalah bahasa pemrograman yang melakukan kompilasi dan dapat dijalankan di berbagai platform UNIX, Windows, dan MacOS [10].

2.10 Library

Library merupakan sekumpulan program yang dapat digunakan pada program lain tanpa terikat satu dengan yang lainnya. Terdapat beberapa *library* yang digunakan dalam melakukan implementasi dalam tugas akhir ini. *Library* yang digunakan dalam bahasa pemrograman Python antara lain; NumPy, Socket, SciPy, PyCrypto, Pandas, dan Rpy2. Sedangkan untuk *library* yang

digunakan pada bahasa pemrograman R yang diakses melalui Rpy2 adalah RegSDC.

2.10.1 NumPy

NumPy adalah *library* Python yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. NumPy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. NumPy bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian [11].

2.10.2 Pandas

Pandas merupakan *library* yang bersifat *open source*, yang mengandung kumpulan alat-alat untuk menunjang proses analisis dan manipulasi data secara cepat, tangguh dan fleksibel. Pandas mendapat popularitas dengan objek *DataFrame*-nya sebagai objek untuk melakukan manipulasi data dengan penomoran yang terintegrasi. Penyajian dan tampilan Pandas dalam pengolahan data, juga menjadi keunggulan Pandas menjadi modul analisis dan manipulasi data [12].

2.10.3 PyCrypto

Modul *Crypto* atau dikenal dengan *PyCrypto*, merupakan modul yang tersedia di Python yang berisikan kumpulan dari fungsi-fungsi *hash* yang aman (seperti SHA256 dan RIPEMD160), dan berbagai algoritma enkripsi (AES, DES, RSA, ElGamal, dll.) [13].

2.10.4 SciPy

SciPy adalah *library* Python gratis dan *open-source* yang digunakan untuk komputasi ilmiah dan komputasi teknis. SciPy berisi modul untuk optimasi, aljabar linier, integrasi, interpolasi, fungsi khusus, FFT, pemrosesan sinyal dan gambar, pemecah ODE, dan tugas-tugas lain yang umum dalam sains dan teknik [14].

2.10.5 Rpy2

Rpy2 adalah antarmuka untuk bahasa pemrograman R yang berjalan tertanam di dalam proses Python. Dengan menggunakan Rpy2 di dalam Python, dapat memungkinkan penggunanya untuk menggunakan komputasi dan mengakses *library* dalam bahasa pemrograman R, tanpa perlu melakukan instalasi R secara manual [15].

2.10.6 RegSDC

RegSDC merupakan *package* yang dapat digunakan di bahasa pemrograman R yang menyediakan alat untuk mempertahankan informasi yang berbasis regresi untuk *statistical disclosure control*. *Package* ini dapat digunakan untuk menghasilkan data mikro yang kontinu sintetis atau hibrida, dan hubungan dengan data aslinya dapat dikendalikan dengan beberapa cara [16].

BAB III
DESAIN

Bab ini menjelaskan tentang perancangan data dan sistem pemungutan suara elektronik yang mengacu pada protokol MVP. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir dan *pseudocode*.

3.1 Perancangan Data

Data asal yang digunakan mengacu pada data yang terdapat pada berkas *primary_result*, yang merupakan hasil rekapitulasi pemungutan suara pada pemilihan di Amerika Serikat pada tahap awal di tahun 2016 yang dijabarkan pada setiap *state* (negara bagian). Berkas tersebut berisikan data *tabular* yang mengandung 24611 baris data, dan memiliki 8 kolom fitur yang terdiri dari nama *state*, singkatan dari *state*, daerah (*county*), kode standar publikasi federal (FIPS), nama partai, nama kandidat, jumlah suara, dan persentase suara. Spesifikasi *dataset* dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi dataset pemilihan umum

Keterangan	**Spesifikasi**
Ekstensi	.csv
Jumlah baris	24611
Jumlah kolom	8

Data pada berkas tersebut mengandung rekapitulasi hasil pemungutan suara pada setiap *state* di Amerika Serikat. Pada pengerjaan ini, dilakukan pemisahan data pada setiap *state*, sehingga akan menghasilkan 49 berkas csv, dimana setiap berkas berisikan data pemilih yang memilih suatu kandidat. Data pemilih dibuat secara sintetis dengan memberikan nomor identifikasi pada setiap suara yang dikumpulkan.

3.1.1 Desain Program Pembuatan Data Uji

Masukan dalam program pembuatan data uji adalah berupa berkas *dataset* awal yaitu *primary_result* yang diunduh dari situs *Kaggle*. Program ini dijalankan secara terpisah dari sistem pemungutan suara, dan berfungsi untuk melakukan pembuatan data uji. Program ini melakukan pengambilan sampel sebanyak N pada setiap *state* yang kemudian dipisah kedalam berkas tersendiri. Desain program tersebut dapat dilihat pada Gambar 3.1.

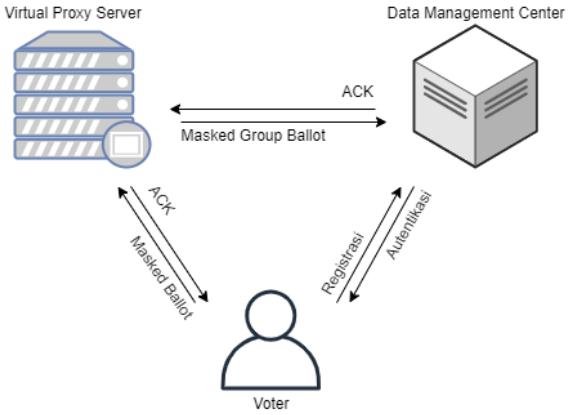
```
   GENERATE-DATA (N, dataPath)
1  f = READ-CSV(dataPath)
2  for each UNIQUE(f.state) do
3      s = SAMPLE(f, N)
4      WRITE-CSV(s)
5  end for
```

Gambar 3.1 Desain Program Pembuatan Data Uji

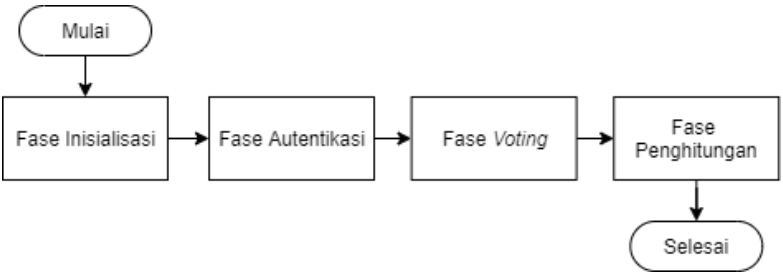
3.2 Desain Umum Sistem

Sistem yang dibuat dalam pengerjaan tugas akhir ini menggunakan arsitektur yang mengacu pada protokol MVP. Di dalam sistem, terdapat tiga entitas, dimana setiap entitas memiliki peran dan fungsinya masing-masing, dan saling berkomunikasi sesuai dengan protokol yang dijalankan. Entitas tersebut adalah DMC (*Data Management Center*), VPS (*Virtual Proxy Server*), dan *voter*. Ilustrasi dari arsitektur yang akan digunakan tertera pada Gambar 3.2.

Implementasi sistem pada tugas akhir ini dilakukan berdasarkan fungsi dan kegunaan pada entitasnya. Oleh karena itu, penjelasan mengenai alur dalam sistem akan dijelaskan terlebih dahulu, kemudian diikuti dengan penjelasan desain masing-masing entitas.



Gambar 3.2 Arsitektur Sistem dalam Protokol MVP



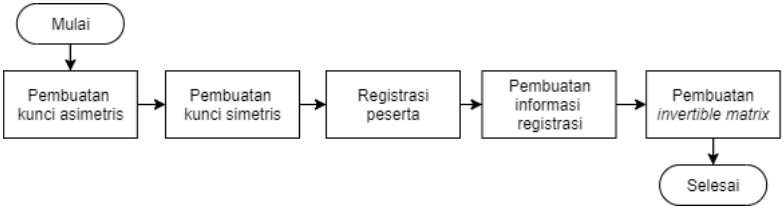
Gambar 3.3 Alur pemungutan suara protokol MVP

Untuk penjelasan detil terkait dengan proses yang terjadi pada setiap fase yang tertera pada Gambar 3.3, akan dijelaskan pada subbab berikut.

3.2.1 Desain Fase Inisialisasi

Fase dimana setiap entitas melakukan penyetelan awal dengan pembuatan kunci, matriks, dan registrasi peserta. Fase inisialisasi diawali dengan pembuatan kunci pada setiap entitas. Untuk setiap entitas, dilakukan pembuatan pasangan kunci asimetris yang

nantinya digunakan untuk enkripsi dengan menggunakan algoritma RSA.



Gambar 3.4 Alur fase inisialisasi

Pembuatan kunci simetris κ yang nantinya digunakan oleh masing-masing entitas, baik untuk pertukaran data dengan AES maupun dengan enkripsi hibrid.

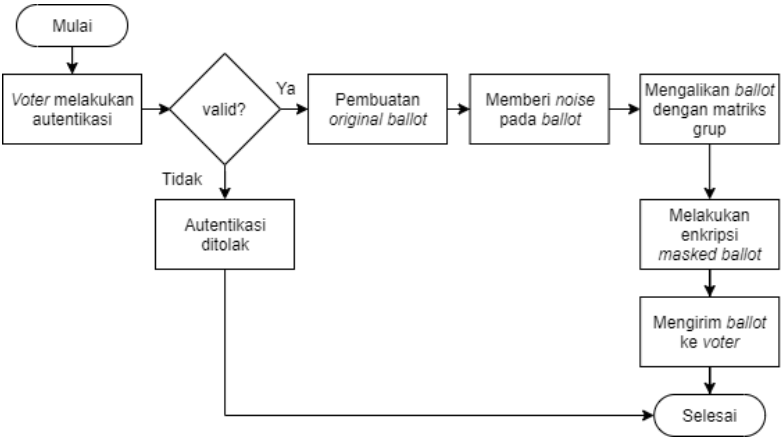
Setiap *voter* v mendaftarkan identitasnya ke DMC (dengan nomor identifikasi), kemudian DMC melakukan generasi informasi registrasi Reg_v dari v yang bersangkutan, dan mengirimkan informasi tersebut ke *voter*. Proses ini dapat dituliskan sebagai: $Setup(Reg_v) \rightarrow (IB_v, GID_v, Index_v)$

Setelah registrasi ditutup, DMC memberikan informasi autentikasi ke VPS $(GID_v, Index_v)$ untuk proses verifikasi *voter* yang *eligible* pada proses pemungutan suara.

Pembuatan *invertible matrix* pada DMC dilakukan setelah proses registrasi peserta selesai. Proses ini dilakukan kepada setiap grup dengan menggunakan s*eed group id* yang bersangkutan.

3.2.2 Desain Fase Autentikasi

Pada fase ini, DMC melakukan proses pembuatan *ballot* (surat suara) dengan mencocokkan identitas *voter* dengan data yang asli, kemudian melakukan enkripsi dan perkalian matriks terhadap *ballot* tersebut dengan faktor identitas *voter*. Rangkaian proses dari fase autentikasi tersedia pada Gambar 3.5.



Gambar 3.5 Alur fase autentikasi

Voter v melakukan autentikasi ke DMC dengan mengirimkan informasi registrasi yang telah diberikan oleh DMC. Pada fase ini, setelah *v* berhasil melakukan autentikasi, DMC akan melakukan pembuatan *ballot*.

Voter v melakukan dekripsi terhadap informasi registrasi terenkripsi dari DMC terlebih dahulu, kemudian mengenkripsi informasi registrasi dengan *private key*-nya.

DMC melakukan dekripsi terhadap informasi registrasi yang dikirimkan *v*, kemudian mencocokan Reg_v ke basis data.

Apabila ditemukan kecocokan antara data registrasi dengan data pada basis data, langkah selanjutnya adalah melakukan pembuatan satu *ballot* untuk *voter* yang sekarang sedang melakukan registrasi.

Langkah awal DMC membuat *satu ballot* adalah dengan membuat *p original ballot*, $\{OB_v^1, \dots, OB_v^j, \dots, OB_v^p\}$, dimana OB_v^j merupakan suara untuk kandidat ke-*j*, dan merupakan vektor biner $1 \times p$, sehingga $OB_v^j[j] = 1$, dan $OB_v^j[i] = 0, \forall i \neq j$.

Langkah selanjutnya adalah dengan menambahkan IB_v ke setiap *original ballot*, sehingga menghasilkan *perturbed ballot* $\{NB_v^1, \dots, NB_v^j, \dots, NB_v^p\}$, dengan $NB_v^j = IB_v + OB_v^j, \forall j \in [1, p]$.

Setelah menghasilkan *perturbed ballot*, diakukan proses masking dengan matriks menurut grup *v*, sehingga menghasilkan *masked ballot* $\{MB_v^1, \dots, MB_v^j, \dots, MB_v^p\}$, dengan $MB_v^j = NB_v^j \times B, \forall j \in [1, p]$.

Langkah selanjutnya adalah melakukan enkripsi setiap *masked ballot* dengan kunci simetris κ untuk membuat *p encrypted masked ballot* $\{EB_v^1, \dots, EB_v^j, \dots, EB_v^p\}$, dengan $EB_v^j = E(\kappa, MB_v^j), \forall j \in [1, p]$.

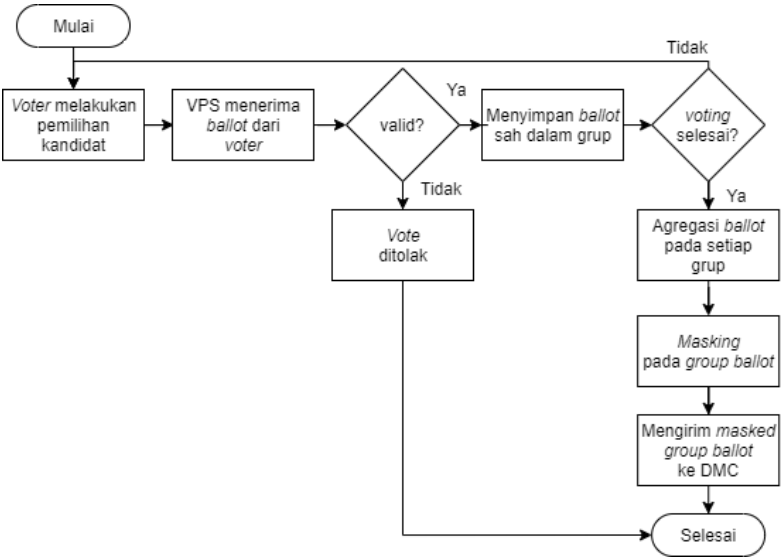
Langkah diatas menghasilkan satu *ballot* yang sah, dan kemudian diberikan kepada *voter* untuk melakukan proses pemilihan.

3.2.3 Desain Fase *Voting*

Pada fase ini, *voter* akan memilih kandidat dan mengirimkan *ballot* yang dipilih ke VPS. VPS akan menerima *ballot* dan mengelompokkannya berdasarkan ID grup. Rangkaian proses dari fase autentikasi tersedia pada Gambar 3.6.

Setiap *voter* memilih kandidat dengan rincian jika *v* memilih kandidat ke-*j*, *v* akan memilih EB_v^j dari M_b, kemudian mengirimkannya melalui message M_2 ke VPS. Dengan $M_2 = E(K_V^+, EB_v | M_a)$.

Setelah VPS menerima *vote* dari *voter*, VPS memverifikasi *vote* yang sah dengan mencocokan $(GID_v, Index_v)$ di basis data VPS, serta melakukan cek apakah *voter* tersebut tidak melakukan pemberian suara ganda dengan mengecek apakah identitas dari *voter* tersebut ada dalam set S. Apabila *vote* tersebut valid, VPS memasukkan identitas ke dalam set S dan menyimpan *vote* tersebut sesuai dengan grupnya. Proses ini berlangsung berulang hingga VPS mendapat tanda bahwa proses pemungutan suara telah berakhir.



Gambar 3.6 Alur fase *voting*

Setelah proses pemungutan suara berakhir, VPS melakukan agregasi *ballot* pada setiap grup, dan menghasilkan *masked group ballot* GB_g untuk setiap grup *voter* g, yang kemudian dikirim ke DMC.

3.2.4 Desain Fase Penghitungan

Setelah VPS menutup entri surat suara, dan menghitung agregasi masing-masing grup. VPS mengirim agregasi tersebut ke DMC untuk mendapatkan hasil pemungutan suara. DMC akan mengalikan masing-masing grup *ballot* dengan invers matriks pada masing-masing grup pada saat awal pembuatan *ballot*. Dengan melakukan agregasi kolom dengan agregasi *noise*, DMC dapat mendapatkan hasil pemungutan suara. Untuk memperoleh agregat dari semua pemilih dalam grup, pertama-tama, DMC mengalikan $A \times GB_g$ dengan B^{-1} untuk mendapatkan *group ballot* G_g. DMC menghitung $colsum\{G_g\}$ dan mengurangkan jumlah *initial ballot noise* dari semua pemilih dalam kelompok g, yang telah

memberikan suara yang sah menurut *S*, i.e.,$\sum_{k=1}^{m} IB_{v_k}$. Sehingga hasil penghitungan akhir dari grup *g* adalah:

$$c_g = colsum\{G_g\} - \sum_{k=1}^{m} IB_{v_k}$$

Pembuktian :

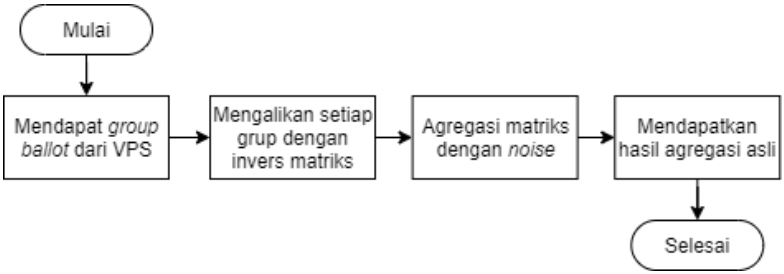
$$c_g = colsum\{G_g\} - \sum_{k=1}^{m} IB_{v_k}$$

$$= colsum\{A \times GB_g \times B^{-1}\} - \sum_{k=1}^{m} IB_{v_k}$$

$$= colsum\{GB_g \times B^{-1}\} - \sum_{k=1}^{m} IB_{v_k}$$

$$= \sum_{k=1}^{m} NB_{v_k} - \sum_{k=1}^{m} IB_{v_k}$$

$$= \sum_{k=1}^{m} OB_{v_k}$$



Gambar 3.7 Alur fase penghitungan

3.3 Desain Entitas DMC

DMC (*Data Management Center*), suatu organisasi atau instansi yang mengadakan kegiatan pemungutan suara. Merupakan entitas penyelenggara dan sebagai pusat dari pemungutan suara, dan bertanggung jawab untuk autentikasi pemilih, menghitung surat suara yang sah, dan pengumuman penghitungan akhir.

3.3.1 Desain Fungsi Pembuatan *Random Invertible Matrix*

Fungsi ini melakukan pembuatan *Random Invertible Matrix* pada setiap grup peserta, kemudian menyimpannya di memori sebagai atribut dari kelas entitas DMC. Setiap matriks yang dihasilkan adalah persegi dan berukuran sesuai banyaknya kandidat.

```
   GENERATE-MAT (size)
1  for each UNIQUE(county) do
2      mat[county] = RANDOM-SQUARE-MAT(size)
3  end for
```

Gambar 3.8 Desain Fungsi Pembuatan Random Invertible Matrix

3.3.2 Desain Fungsi Server Soket TCP

Fungsi ini melakukan inisiasi server soket TCP pada DMC yang digunakan untuk berkomunikasi dengan entitas lainnya. Fungsi ini bertujuan sebagai fungsi *threading* yang merupakan *server* dari entitas VPS dan *voter*.

```
   LISTEN-DMC-TCP-SOCKET (IP, PORT)
1  sock = TCP-SOCKET(IP, PORT)
2  while True do
3      ACCEPT-CONNECTION(sock)
4      M = GET-MESSAGE(sock)
5      if M = 'VPS' then
6          START-THREAD(VPS-THREAD)
7      else if M = 'VOTER' then
8          START-THREAD(VOTER-THREAD)
9      end if
10 end while
```

Gambar 3.9 Desain fungsi *server* soket TCP untuk DMC

3.3.3 Desain Fungsi *Thread* untuk *VPS*

Threading diperlukan agar pemrosesan dapat dilakukan secara paralel dengan hanya menjalankan satu program saja. Fungsi *thread* ini bertujuan untuk melayani dan menerjemahkan

perintah dari VPS terkait dengan proses maupun aliran data dalam DMC.

```
   VPS-THREAD ()
1  let client be VPS
2  while client is True:
3      data = message from client
4      if data = symkey exchange do
5          GENERATE-SYMKEY(client)
6      else if data = pubkey exchange do
7          EXCHANGE-PUBKEY(client)
8      else if data = get candidate do
9          Send (client, candidate)
10     else if data = get group list do
11         Send (client, group_list)
12     else if data = get voter regist do
13         EXPORT-VOTER-INDEX()
14     else if data = voting result do
15         COUNT-VOTES(client)
16     else
17         OUTPUT('Disconnected')
18     end if
19 end while
```

Gambar 3.10 Desain fungsi *thread* VPS

3.3.4 Desain Fungsi *Thread* untuk *Voter*

Fungsi *thread* ini bertujuan untuk melayani dan menerjemahkan perintah dari entitas *voter* terkait dengan proses maupun aliran data dalam DMC.

```
    VOTER-THREAD ()
1  let client be voter
2  while client is True:
3      data = Recv from client
4      if data = pubkey exchange do
5          EXCHANGE-PUBKEY(client)
```

Gambar 3.11 Desain fungsi *thread voter* (a)

```
6       else if data = voter registration do
7           VOTER-REGISTRATION(client)
8       else if data = voter authentication do
9           VOTER-AUTH(client)
10      else
11          OUTPUT('Disconnected')
12      end if
13  end while
```

Gambar 3.12 Desain fungsi *thread voter* (b)

3.3.5 Desain Fungsi Registrasi *Voter*

Fungsi ini digunakan oleh DMC untuk melakukan proses registrasi *voter*. Masukan dari fungsi ini adalah informasi identitas dari *voter*. Apabila *voter* mengirimkan pesan registrasi, pesan tersebut akan disimpan oleh DMC, kemudian DMC mengirimkan pesan berupa informasi registrasi kepada *voter* yang menandakan bahwa *voter* tersebut berhasil didaftarkan. Desain fungsi ini dapat dilihat pada Gambar 3.13.

```
    VOTER-REGISTRATION ()
1   let client be voter
2   let participant, regdb, idreg be dict
3   let regv be uuid
4   data = Recv from client
5   insert data.id to participant[data.county]
6   M1 = (RND_MAT, data.county, data.index)
7   regdb[regv] = M1
8   Send ENCRYPT-RSA(regv)
```

Gambar 3.13 Desain fungsi registrasi *voter*

3.3.6 Desain Fungsi Autentikasi *Voter*

Fungsi ini digunakan oleh DMC untuk melakukan proses autentikasi *voter* pada saat pemungutan suara dimulai. Fungsi menerima registrasi *voter* yang valid, yang kemudian dijadikan basis pembuatan *ballot* yang sah dikeluarkan oleh DMC. Desain fungsi ini dapat dilihat pada Gambar 3.14.

```
   VOTER-AUTH ()
1  let client be voter
2  let candidates be list of candidates
3  data = Recv from client
4  regv = DECRYPT-RSA(data)
5  if regv is in regdb do
6      Ib_v, GID_v, index_v = regdb[regv]
7      M_a = AES-ENCRYPT((gid_v, index_v)
8      let M_c be dict
9      let Ob_v be identity matrix
10     for each c in candidates do
11         Nb_v = Ob_v[c.index] + Ib_v
12         Mb_v = MATMUL(Nb_v, mat_g[GID_v])
13         Eb_v = AES-ENCRYPT(Mb_v)
14         M_c[c] = Eb_v
15     end for
16     M1 = (M_a, M_c)
17     SEND M_1
18 end if
```

Gambar 3.14 Desain fungsi autentikasi *voter* pada DMC

3.3.7 Desain Fungsi Rekap Pemungutan Suara

Fungsi ini dipakai pada saat proses pemungutan suara telah berakhir dan bertujuan untuk mengetahui hasil dari pemungutan suara. Masukan pada fungsi ini adalah berupa *masked group ballot* yang dikirimkan oleh VPS. Karena pada pasalnya *masked group ballot* merupakan sekumpulan *masked ballot* pada setiap grup, untuk mendapatkan rekapitulasi hasil dari pemungutan suara, pada fungsi ini akan dilakukan perkalian matriks terhadap setiap *masked ballot* pada setiap grup dengan invers matriks grup yang bersangkutan. Kemudian dilakukan proses agregasi kolom pada setiap matriks untuk mendapatkan hasil pemungutan suara yang sesungguhnya. Desain fungsi ini dapat dilihat pada Gambar 3.15.

```
   COUNT-VOTES ()
1  let client be VPS
2  let result be vector of zeroes
3  data = recv from client
4  Gb_g = AES-DECRYPT(data)
5  for each Gb in Gb_g do
6      Agg = COLSUM(Gb)
7      Cg = MATMUL(agg, mat_g[Gb.name])
8      Cg = Cg - SUM(ib_v[Gb.name])
9      result = result + Cg
10 end for
11 OUTPUT(result)
12 OUTPUT(VOTING ENDED)
```

Gambar 3.15 Desain fungsi rekapitulasi hasil pemungutan suara

3.4 Desain Entitas VPS

VPS (*Virtual Proxy Server*) merupakan entitas yang memiliki tanggung jawab untuk mengumpulkan surat suara yang sah dari pemilih yang telah melakukan autentikasi, dan meneruskan *group masked ballot* ke DMC.

3.4.1 Desain Fungsi Server Soket TCP

Fungsi ini melakukan inisiasi server soket TCP pada VPS yang digunakan untuk berkomunikasi dengan entitas *voter*. Desain fungsi ini dapat dilihat pada Gambar 3.16.

```
   LISTEN-VPS-TCP-SOCKET (IP, PORT)
1  sock = TCP-SOCKET(IP, PORT)
2  while True do
3      ACCEPT-CONNECTION(sock)
4      M = GET-MESSAGE(sock)
5      if M = 'VOTER' then
6          START-THREAD(VOTER-THREAD)
7      end if
8  end while
```

Gambar 3.16 Desain fungsi *server* soket TCP untuk DMC

3.4.2 Desain Fungsi *Thread* untuk *Voter*

Fungsi ini bertujuan sebagai fungsi *threading* yang merupakan *server* dari entitas *voter*. Desain fungsi ini dapat dilihat pada Gambar 3.17.

```
   VOTER-THREAD ()
1  let client be voter
2  while client is True:
3      data = Recv from client
4      if data = pubkey exchange do
5          EXCHANGE-PUBKEY(client)
6      else if data = end of registration do
7          FETCH-REG-DB()
8      else if data = vote do
9          HANDLE-VOTE(client)
10     else
11         OUTPUT('Disconnected')
12     end if
13 end while
```

Gambar 3.17 Desain fungsi *thread voter* pada VPS

3.4.3 Desain Fungsi Impor Data Registrasi

Pada fungsi ini VPS akan melakukan *import* terhadap data registrasi yang dihasilkan oleh DMC. Data tersebut digunakan oleh VPS untuk melakukan verifikasi terhadap *ballot* yang dikirimkan oleh *voter*. Desain fungsi ini dapat dilihat pada Gambar 3.18.

```
   FETCH-REG-DB ()
1  let server be DMC
2  SEND 'Reg flag'
3  READ participant_file to memory
4  READ id_reg_file to memory
```

Gambar 3.18 Desain fungsi impor data registrasi pada VPS

3.4.4 Desain Fungsi *Vote Handler*

Fungsi ini bertujuan untuk menerima dan melakukan validasi *ballot* yang telah dipilih dari *voter*, dan mengumpulkan *ballot* yang valid ke dalam memori, sehingga hanya *ballot* yang valid saja yang disimpan. Masukan dari fungsi ini berupa *ballot* yang telah dipilih dan dikirimkan oleh *voter*. Desain fungsi ini dapat dilihat pada Gambar 3.19.

```
   VOTER-THREAD ()
1  let client be voter
2  data = Recv from client
3  ballot, Mb = FETCH(data)
4  GID_v, index_v = FETCH(AES-DECRYPT(Mb))
5  if participant[GID_v][index_v} exists do
6      if (GID_v, index_v) not in voted_set do
7          INSERT (GID_v, index_v) to voted_set
8          INSERT ballot to GB_g[GID_V]
9      end if
10 end if
```

Gambar 3.19 Desain fungsi *vote handler* pada VPS

3.4.5 Desain Fungsi Pembuatan Masked Group Ballot

Fungsi ini melakukan agregasi *ballot* dan *masking* pada setiap grupnya. Fungsi ini dijalankan ketika proses pemungutan suara telah berakhir. Desain fungsi ini dapat dilihat pada Gambar 3.20.

```
   AGGREGATE-VOTES ()
1  let c be LEN(candidates)
2  for each ballots in Gb do
3      agg = SUM(Gb)
4      mat_noise = RANDOM_MAT(c-1, c)
5      sum_noise = COLSUM(mat_noise)
6      agg = APPEND(mat_noise, agg - sum_noise)
7      Gb_g[ballots.key] = ROMM(agg, rnd_mat, p)
8  end for
```

Gambar 3.20 Desain fungsi pembuatan *masked group ballot* di VPS

3.4.6 Desain Fungsi Masking

Fungsi ini memiliki masukan berupa matriks persegi, yang kemudian diterapkan *Statistical Disclosure Limitation* berupa *Random Orthogonal Matrix Masking* yang telah dijelaskan pada subbab 2.4.1. Keluaran dari fungsi ini berupa matriks persegi yang berukuran sama, namun nilai dari matriks tersebut merupakan hasil dari *masking*. Desain fungsi ini dapat dilihat pada Gambar 3.21.

```
   ROMM (X, Y, p)
1  Assign X, Y, as R Matrix
2  Mb = RegSDCRomm(X,p,Y)
3  return ASARRAY(Mb)
```

Gambar 3.21 Desain fungsi *masking*

3.5 Desain Entitas Voter

Voter merupakan entitas yang melakukan autentikasi serta melakukan pemilihan suara, dari surat suara yang diberikan oleh *server*.

3.5.1 Desain Fungsi Registrasi

Fungsi ini bertujuan untuk melakukan registrasi sebagai peserta pemungutan suara dengan melakukan pembuatan identitas secara sintetis yang mengacu pada *dataset*. Data registrasi yang dihasilkan kemudian dikirimkan ke DMC untuk memperoleh informasi registrasi yang sah dari DMC dan disimpan satu persatu ke memori. Desain fungsi ini dapat dilihat pada Gambar 3.22.

```
   REGISTRATION ()
1  let server be DMC
2  df = READ-CSV(dataset)
3  for each voter_data in df do
4     SEND (county, voter_id)
5     regv = Recv from server
6  end for
```

Gambar 3.22 Desain fungsi registrasi *voter*

3.5.2 Desain Fungsi Pemilihan Kandidat

Pada fungsi ini dilakukan proses autentikasi dan pemilihan kandidat oleh *voter*. Entitas *voter* akan satu persatu melakukan iterasi pada informasi registrasi yang tersimpan. Pada setiap iterasi, satu informasi registrasi *voter* akan dikirim ke DMC untuk mendapatkan *ballot* yang sah. Setelah DMC mengirimkan *ballot* ke *voter*, dilakukan pemilihan kandidat oleh *voter* sesuai dengan kandidat yang dipilih pada *dataset*. Desain fungsi ini dapat dilihat pada

```
   REGISTRATION ()
1  let server be VPS
2  for each voter in voters do
3      data = Recv from DMC
4      Ma, Mb = FETCH(data)
5      selected = Mb[voter_choice]
6      SEND (selected, Ma)
7  end for
```

Gambar 3.23 Desain fungsi pemilihan kandidat oleh *voter*

3.6 Desain Fungsi Tambahan

Fungsi tambahan ini merupakan kumpulan modul dan fungsi yang digunakan oleh ketiga entitas. Karena pada pasalnya terdapat beberapa fungsi yang sama pada ketiga entitas tersebut, sehingga tujuan dari pembuatan fungsi tambahan ini adalah untuk mengurangi pengulangan penulisan kode sumber pada masing-masing entitas.

[*Halaman ini sengaja dikosongkan*]

.

BAB IV
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Implementasi ditulis menurut entitas yang terlibat dalam sistem.

4.1 Lingkungan Implementasi

Dalam melakukan implementasi terhadap simulasi protokol pemungutan suara elektronik yang mengacu pada protokol MVP, diperlukan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan *laptop personal computer* (PC). PC yang digunakan memiliki spesifikasi prosesor Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), *Random Access Memory* (RAM) sebesar 8 GB.

4.1.2 Perangkat Lunak

PC dari sisi perangkat lunak memiliki spesifikasi antara lain menggunakan sistem operasi Windows 10 64-bit dengan lingkungan pengembangan menggunakan bahasa pemrograman Python 3.6, dan dilengkapi dengan *library* antara lain NumPy, Pandas, SciPy, sklearn, dan rpy2. Untuk antarmuka rpy2 dilengkapi dengan *package* R berupa RegSDC.

4.2 Implementasi Program Pembuatan Data Uji

Subbab ini menjelaskan implementasi program pembuatan data uji yang nantinya akan menjadi masukan ke dalam sistem. Implementasi dilakukan merujuk pada subbab 3.1. Terdapat dua implementasi dalam program pembuatan data uji. Implementasi pertama yang dapat dilihat pada Kode Sumber 4.1 berfungsi melakukan pemetaan nomor identitas peserta pada *dataset* awal di setiap *state*.

```
   def sampling():
1  df = pd.read_csv(DATASET_PATH)
2  state_unique = df['state'].unique()
3  id_counter = 1
4  df_columns = ['state_abbr', 'id', 'county',
   'party', 'candidate_chosen']
5  for s in state_unique:
6      df_s = df[df['state'] == s]
7      d = []
8      for i in range(len(df_s)):
9          c = df_s.iloc[i]
10         for _ in range(c['votes']):
11             d.append([
12                 c['state_abbreviation'],
13                 id_counter,
14                 c['county'],
15                 c['party'],
16                 c['candidate'],
17             ])
18             id_counter += 1
19     df = pd.DataFrame(d, columns=df_columns)
20     df.to_csv(OUTPUT_FOLDER_PATH + str(s) +
   '.csv', index=False)
```

Kode Sumber 4.1 Implementasi pemetaan nomor identitas peserta

Implementasi pada Kode Sumber 4.2 berfungsi melakukan pengambilan sampel dari hasil implementasi Kode Sumber 4.1 pada masing-masing *state* .

```
   def sampling():
1  df_primary = pd.read_csv(DATASET_PATH)
2  for c in df_primary['state'].unique():
3      df = pd.read_csv(DATASET_PATH_2 + c +
   '.csv')
4      for n in NUM_SAMPLE:
5          df2 = df.sample(n=n, random_state=0)
6          df2.to_csv(OUTPUT_FOLDER_PATH+str(n)
   + '/' + c + '.csv', index=False)
```

Kode Sumber 4.2 Implementasi pengambilan sampel setiap *state*

4.3 Implementasi Entitas DMC

Subbab ini menjelaskan implementasi program dari entitas *Data Management Center* (DMC) yang berperan sebagai penyelenggara pemungutan suara. Implementasi dilakukan merujuk pada subbab 3.3.

4.3.1 Implementasi Fungsi Pembuatan *Random Invertible Matrix*

Subbab ini berisi implentasi dari fungsi pembuatan *random invertible matrix* dengan merujuk pada subbab 3.3.1.

```
   def keygen (self):
1  for key in self.group_list:
2      self.group_invertible_matrices[key] =
   make_spd_matrix(len(self.candidates))
```

Kode Sumber 4.3 Implementasi fungsi pembuatan matriks acak

4.3.2 Implementasi Fungsi Server Soket TCP

Fungsi ini melakukan inisiasi server soket TCP pada DMC. Fungsi ini merujuk pada subbab 3.3.2.

```
   def listen_dmc_tcp_socket(self)
1  self.sock.listen(2)
2  print("DMC is listening at HOST " +
   repr(self.DMC_HOST) + " and PORT " +
   repr(self.DMC_PORT))
3  while True:
4      client, addr = self.sock.accept()
5      data = recv_msg(client)
6      if data == b'VPS':
7          send_only(client, b'OK')
8
   threading.Thread(target=self.vps_thread,
   args=(client,addr)).start()
9      elif data == b'VOTER':
10         send_only(client, b'OK')
```

Kode Sumber 4.4 Implementasi fungsi server soket TCP pada DMC (a)

```
11
   threading.Thread(target=self.voter_thread,
   args=(client,addr)).start()
12     else:
13         client.close()
14 self.sock.close()
```

Kode Sumber 4.5 Implementasi fungsi server soket TCP pada DMC (b)

4.3.3 Implementasi Fungsi *Thread* untuk VPS

Subbab ini berisi implementasi dari fungsi *threading* yang dipakai DMC untuk melayani VPS. Fungsi ini merujuk pada subbab 3.3.3.

```
   def vps_thread (self, client, address)
15 while True:
16     try:
17         data = recv_msg(client)
18         if data:
19             if data == b'GET_SYMKEY':
20                 self.sym_key =
   generate_symmetric_key(client,
   self.password_sym_key, self.salt)
21             elif data == b'PUBKEY_VPS':
22                 send_only(client, b'OK')
23                 exchange_public_key(client,
   'vps-key.pem', BASE_FOLDER_PATH,
   RSA_PUBLIC_KEY)
24             elif data == b'GET_CANDIDATES':
25                 send_stream(client,
   pickle.dumps(self.candidates))
26             elif data == b'GET_GROUP_LIST':
27                 send_stream(client,
   pickle.dumps(self.group_list))
28             elif data == b'FETCH_REG':
29
   self.export_voter_index(client)
30             elif data == b'VOTES':
```

Kode Sumber 4.6 Implementasi fungsi server soket TCP untuk VPS (a)

```
31                  send_only(client, b'OK')
32                  self.count_votes(client)
33              elif data ==
   b'VOTED_PARTICIPANT':
34                  self.fetch_set_voted(client)
35      except:
36          client.close()
37          return False
```

Kode Sumber 4.7 Implementasi fungsi server soket TCP untuk VPS (b)

4.3.4 Implementasi Fungsi *Thread* untuk *Voter*

Subbab ini berisi implementasi dari fungsi *threading* yang dipakai DMC untuk melayani *voter*. Fungsi ini merujuk pada subbab 3.3.4.

```
   def voter_thread (self, client, address):
1  while True:
2      try:
3          data = recv_msg(client)
4          if data:
5              if data == b'PUBKEY_VOTER':
6                  send_only(client, b'OK')
7                  exchange_public_key(client,
   'voter-key.pem', BASE_FOLDER_PATH,
   RSA_PUBLIC_KEY)
8              elif data == b'VOTER_REG':
9                  send_only(client, b'OK')
10                 self.registration(client)
11             elif data == b'AUTH':
12                 send_only(client, b'OK')
13                 self.handle_auth(client)
14     except Exception as e:
15         client.close()
16         return False
```

Kode Sumber 4.8 Implementasi *thread voter* pada DMC

4.3.5 Implementasi Fungsi Registrasi *Voter*

Subbab ini berisi implementasi dari fungsi registrasi peserta yang pada DMC. Fungsi ini merujuk pada subbab 3.3.5.

```
   def voter_thread (self, client, address):
1  data = recv_msg(client)
2  while data != b'END':
3      try:
4          county, voter_id =
   pickle.loads(data)
5          print(county, voter_id)
6          regv = ''
7          while True:
8              regv = str(uuid.uuid4())
9              if regv not in
   self.reg_db.keys(): break
10
   self.participant[county].append(voter_id)
11         self.reg_db[regv] =
   (np.random.randint(0, 10,
   size=len(self.candidates)), regv, county,
   len(self.participant[county]) - 1)
12         self.id_reg[voter_id] = regv
13         data = send_msg(client,
   encrypt_rsa('dmc', 'public', regv,
   BASE_FOLDER_PATH))
14     except Exception as e:
15         print(e)
```

Kode Sumber 4.9 Implementasi fungsi registrasi *voter* pada DMC

4.3.6 Implementasi Fungsi Autentikasi Voter

Subbab ini berisi implementasi dari fungsi autentikasi peserta untuk dapat mendapatkan surat suara yang sah dari DMC. Fungsi ini merujuk pada subbab 3.3.6.

```
   def voter_thread (self, client, address):
1  encrypted_reg_v = recv_stream(client)
2  reg_v = decrypt_rsa('dmc', 'private',
   encrypted_reg_v, BASE_FOLDER_PATH)
3  if reg_v in self.reg_db:
4      send_only(client, b'OK') # ACK
5      # i
6      ib_v, cert_v, gid_v, index_v =
   self.reg_db[reg_v]
7      # ii
8      m_a = (gid_v, index_v)
9      concated_msg = concat_msg((cert_v,
   gid_v, index_v))
10     m_b = aes_encrypt(concated_msg,
   self.sym_key)
11     # iii-vii
12     m_c = {}
13     for x in range(len(self.candidates)):
14         nb_v =
   np.add(self.original_ballot[x], ib_v)
15         mb_v =
   nb_v.dot(self.group_invertible_matrices[gid_
   v])
16         eb_v = aes_encrypt(concat_msg(mb_v),
   self.sym_key)
17         m_c[self.candidates[x]] = eb_v
18     m_1 = (m_a, m_b, m_c)
19     send_stream(client, pickle.dumps(m_1))
20     print("SENT", (gid_v, index_v))
21 else:
22     send_only(client, b'FORBIDDEN')
```

Kode Sumber 4.10 Implementasi fungsi autentikasi peserta

4.3.7 Implementasi Fungsi Rekap Pemungutan Suara

Subbab ini berisi implementasi dari fungsi rekap pemungutan suara untuk dapat mendapatkan hasil dari proses pemungutan suara. Fungsi ini merujuk pada subbab 3.3.7.

```
   def listen_vps_tcp_socket (self):
1  msg = recv_stream(client)
2  group_collection =
   fetch_msg(aes_decrypt(msg, self.sym_key))
3  final_count = np.zeros(len(self.candidates))
4  for key in group_collection.keys():
5      if not isinstance(group_collection[key],
   (np.ndarray)):
6          continue
7      group_noise =
   np.zeros(len(self.candidates))
8      for v_idx in
   range(len(self.participant[key])):
9          v_id = self.participant[key][v_idx]
10         if (key, v_idx) in
   self.voted_participants:
11             reg_v = self.id_reg[v_id]
12             ib_v, a, b, c =
   self.reg_db[reg_v]
13             group_noise =
   np.add(group_noise, ib_v)
14     demasked =
   np.array(group_collection[key]).dot(np.linal
   g.inv(self.group_invertible_matrices[key]))
15     colsum_demasked = np.sum(demasked,
   axis=0)
16     count = np.subtract(colsum_demasked,
   group_noise)
17     final_count = np.add(final_count, count)
18     print("RESULT", final_count)
```

Kode Sumber 4.11 Implementasi fungsi rekap pemungutan suara

4.4 Implementasi Entitas VPS

Subbab ini menjelaskan implementasi program dari entitas *Virtual Proxy Server* (VPS) yang berperan sebagai pihak perantara dalam pemungutan suara. Implementasi dilakukan merujuk pada subbab 3.4.

4.4.1 Implementasi Fungsi Server Soket TCP

Fungsi ini melakukan inisiasi server soket TCP pada VPS yang digunakan untuk berkomunikasi dengan entitas lainnya.

```
   def listen_vps_tcp_socket (self):
1  self.sock.listen(1)
2  print("VPS is listening at HOST " + repr(sel
   f.VPS_HOST) +
   " and PORT " + repr(self.VPS_PORT))
3  client, addr = self.sock.accept()
4  print('Connected to :', addr[0], ':', addr[1
   ])
5  data = recv_msg(client)
6  if data == b'VOTER':
7      send_only(client, b'OK')
8      self.thread_vps_voter = threading.Thread
   (
9          target=self.voter_thread, args=(clie
   nt, addr))
10     self.thread_vps_voter.start()
11 else:
12     client.close()
```

Kode Sumber 4.12 Implementasi server soket TCP untuk VPS

4.4.2 Implementasi Fungsi *Thread* untuk *Voter*

Subbab ini berisi implementasi dari fungsi *threading* yang dipakai VPS untuk melayani *voter*.

```
   def voter_thread (self, client, address):
1  while True:
2      try:
3          data = recv_msg(client)
4          if data:
5              if data == b'PUBKEY_VOTER':
6                  send_only(client, b'OK')
7                  self.exchange_public_key
   (client, 'voter-key.pem')
```

Kode Sumber 4.13 Implementasi *thread voter* pada VPS (a)

```
8                elif data == b'REG_END':
9                    self.fetch_reg_db()
10               elif data == b'VOTE':
11                   send_only(client, b'OK')
12                   self.handle_vote(client)
13           else:
14               raise error('Disconnected.')
15       except:
16           client.close()
17           return False
```

Kode Sumber 4.14 Implementasi *thread voter* pada VPS (b)

4.4.3 Implementasi Fungsi Impor Data Registrasi

Pada fungsi ini VPS akan melakukan *import* terhadap data registrasi yang dihasilkan oleh DMC. Data tersebut digunakan oleh VPS untuk melakukan verifikasi terhadap *ballot* yang dikirimkan oleh *voter*.

```
   def fetch_reg_db(self):
1  send_msg(self.dmc_sock, b'FETCH_REG')
2  with open('participant.pickle', 'rb') as
   handle:
3      self.participant = pickle.load(handle)
4  with open('id_reg.pickle', 'rb') as handle:
5      self.id_reg = pickle.load(handle)
```

Kode Sumber 4.15 Implementasi impor data registrasi *voter*

4.4.4 Implementasi Fungsi *Vote Handler*

Fungsi ini bertujuan untuk menerima dan melakukan validasi *ballot* yang telah dipilih dari *voter*, dan mengumpulkan *ballot* yang valid ke dalam memori, sehingga hanya *ballot* yang valid saja yang disimpan. Masukan dari fungsi ini berupa *ballot* yang telah dipilih dan dikirimkan oleh *voter*.

```
   def handle_vote(self, client):
1  m_2 = recv_stream(client)
```

Kode Sumber 4.16 Implementasi *vote handler* (a)

```
2  selected_ballot, m_b = self.fetch_msg(m_2)
3  cert_v, gid_v, index_v = self.fetch_msg(
4      self.aes_decrypt(m_b, self.sym_key))
5  if
   self.id_reg[self.participant[gid_v][index_v]
   ] == cert_v:
6      if len(self.voted) !=
   (self.voted.add((gid_v, index_v)) or
   len(self.voted)):
7          ballot =
   np.array(self.fetch_msg(self.aes_decrypt(sel
   ected_ballot, self.sym_key)))
8       self.group_ballot[gid_v].append(ballot)
9  send_only(client, b'OK')
```

Kode Sumber 4.17 Implementasi *vote handler* (b)

4.4.5 Implementasi Fungsi Pembuatan *Masked Group Ballot*

Fungsi ini melakukan agregasi *ballot* dan *masking* pada setiap grupnya. Fungsi ini dijalankan ketika proses pemungutan suara telah berakhir.

```
   def generate_gb_g(self):
1  for group_name in self.group_list:
2      gb_g =
   np.random.rand(len(self.candidates) - 1,
   len(self.candidates))
3      target =
   self.group_aggregate[group_name] -
   np.array(gb_g).sum(axis=0)
4      gb_g_appended = np.append(gb_g,
   [target], axis=0)
5      self.gb_g[group_name] =
   self.romm(gb_g_appended, 10)
```

Kode Sumber 4.18 Implementasi pembuatan *masked group ballot*

4.4.6 Implementasi Fungsi *Masking*

Fungsi ini memiliki masukan berupa matriks persegi, yang kemudian diterapkan *Statistical Disclosure Limitation* berupa

Random Orthogonal Matrix Masking yang telah dijelaskan pada subbab 2.x. Keluaran dari fungsi ini berupa matriks persegi yang berukuran sama, namun nilai dari matriks tersebut merupakan hasil dari *masking*.

```
   def generate_gb_g(self):
1  nr, nc = B.shape
2  nrx, ncx = X.shape
3  Br = ro.r.matrix(B, nrow=nr, ncol=nc)
4  Xr = ro.r.matrix(X, nrow=nrx, ncol=ncx)
5  ro.r.assign("y", Br)
6  ro.r.assign("x", Xr)
7  r("MB <- RegSDCromm(y, "+str(p)+", x)")
8  return np.asarray(r('MB'))
```

Kode Sumber 4.19 Implementasi *random orthogonal matrix masking*

4.5 Implementasi Entitas Voter

Subbab ini menjelaskan implementasi program dari entitas *voter* yang berperan sebagai peserta dalam pemungutan suara. Implementasi dilakukan merujuk pada subbab 3.5.

4.5.1 Implementasi Fungsi Registrasi Peserta

Subbab ini menjelaskan implementasi program dari entitas *voter* yang melakukan registrasi ke DMC sebelum proses pemungutan suara berlangsung. Implementasi dilakukan merujuk pada subbab 3.5.1.

```
   def registration(self):
1  send_msg(self.dmc_sock, b'VOTER_REG')
2      for i in range(len(self.df)):
3          data = self.df.iloc[i]
4          msg = pickle.dumps((data['county'],
   data['id']))
5          resp = send_msg(self.dmc_sock, msg)
6          self.reg_data[data['id']] = (resp,
   data['candidate_chosen'])
```

Kode Sumber 4.20 Implementasi fungsi registrasi peserta (a)

```
7       send_only(self.dmc_sock, b'END')
    send_only(self.vps_sock, b'REG_END')
```

Kode Sumber 4.21 Implementasi fungsi registrasi peserta (b)

4.5.2 Implementasi Fungsi Pemilihan Kandidat

Subbab ini menjelaskan implementasi program dari entitas *voter* yang melakukan pemilihan kandidat pada saat proses pemungutan suara berlangsung. Surat suara diperoleh dari DMC, dan kemudian dikirimkan ke VPS setelah *voter* melakukan pemilihan kandidat. Implementasi dilakukan merujuk pada subbab 3.5.2.

```
   def auth_and_vote(self):
1  voters = [x for x in self.reg_data.keys()]
2  for voter in voters:
3      send_msg(self.dmc_sock, b'AUTH')
4      enc_regv, candidate_chosen =
   self.reg_data[voter]
5      send_stream(self.dmc_sock, enc_regv)
6      res = recv_msg(self.dmc_sock)
7      if res != b'FORBIDDEN':
8          msg = recv_stream(self.dmc_sock)
9          m_1 = pickle.loads(msg)
10         m_a, m_b, m_c = m_1
11         selected_ballot =
   m_c[candidate_chosen]
12         m_2 = concat_msg((selected_ballot,
   m_b))
13         send_msg(self.vps_sock, b'VOTE')
14         send_stream(self.vps_sock,
   m_2.encode())
15         recv_msg(self.vps_sock)
16         print("Submitted", voter)
```

Kode Sumber 4.22 Implementasi fungsi pemilihan kandidat

4.6 Implementasi Fungsi Tambahan

Subbab ini menjelaskan implementasi fungsi-fungsi pelengkap tambahan yang dipakai dalam ketiga entitas. Fungsi-fungsi tersebut meliputi metode komunikasi, pertukaran kunci, enkapsulasi data, dan fungsi *helper* lainnya. Implementasi dilakukan merujuk pada subbab 3.6.

BAB V
UJI COBA DAN EVALUASI

Bab ini akan membahas mengenai hasil uji coba sistem yang telah diimplementasikan. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah *laptop personal computer* (PC). Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi prosesor Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), *Random Access Memory* (RAM) sebesar 8 GB.

5.2 Deskripsi *Dataset*

Pada tugas akhir ini, data yang digunakan sebagai masukan awal dari sistem adalah hasil dari program pembuatan data uji pada subbab 3.1 yang berupa hasil *sampling* dari peserta pemilihan umum pada setiap *state*.

5.3 Skenario Pengujian

Pengujian yang dilakukan adalah pengujian lokal, dimana implementasi akan dijalankan dan dilakukan evaluasi di lingkungan uji coba pada subbab 5.1. Masukan dari pengujian adalah *dataset* yang telah dibuat dari keluaran program pembuatan data uji. Ada dua jenis uji coba yang dilakukan yaitu uji coba kebutuhan sistem dan uji coba kinerja sistem.

5.4 Uji Coba Kebenaran Sistem

Kebenaran akan sistem pemungutan suara elektronik akan dibuktikan pada subbab ini, dengan menguji dan melakukan simulasi apakah kebutuhan pada sistem ini sudah terpenuhi.

Uji coba dilakukan dengan mengecek kebenaran pada data masukan dengan data keluaran hasil program. Pengecekan dilakukan dengan membandingkan nilai pada *dataset* dengan nilai

hasil dari sistem yang diujikan. Pengujian akan dilakukan pada setiap *state* dengan masukan berupa data hasil *sampling* dengan suatu *seed* tertentu. Penggunaan *seed* bertujuan supaya data yang didapatkan adalah konsisten, dan dapat dilakukan reproduksi ulang. Pengujian setiap *state* dengan ukuran data masing-masing 50 data dapat dilihat pada Tabel 5.1-Tabel 5.2.

Tabel 5.1 Hasil uji kebenaran sistem (a)

Nama *State*	Jumlah Kandidat	Jumlah Data Uji	Jumlah Data Benar
Alabama	7	50	50
Alaska	7	50	50
Arizona	5	50	50
Arkansas	6	50	50
California	5	50	50
Colorado	2	50	50
Connecticut	5	50	50
Delaware	5	50	50
Florida	6	50	50
Georgia	7	50	50
Hawaii	6	50	50
Idaho	5	50	50
Illinois	6	50	50
Indiana	5	50	50
Iowa	10	50	50
Kansas	6	50	50
Kentucky	6	50	50
Louisiana	6	50	50
Maine	2	50	50
Maryland	5	50	50
Massachusetts	6	50	50
Michigan	6	50	50
Mississippi	6	50	50
Missouri	6	50	50

Tabel 5.2 Hasil uji kebenaran sistem (b)

Nama *State*	Jumlah Kandidat	Jumlah Data Uji	Jumlah Data Benar
Montana	5	50	50
Nebraska	5	50	50
Nevada	7	50	50
New Hampshire	8	50	50
New Jersey	5	50	50
New Mexico	5	50	50
New York	5	50	50
North Carolina	6	50	50
North Dakota	2	50	50
Ohio	6	50	50
Oklahoma	6	50	50
Oregon	5	50	50
Pennsylvania	5	50	50
Rhode Island	5	50	50
South Carolina	8	50	50
South Dakota	5	50	50
Tennessee	7	50	50
Texas	7	50	50
Utah	4	50	50
Vermont	6	50	50
Virginia	7	50	50
Washington	5	50	50
West Virginia	5	50	50
Wisconsin	5	50	50
Wyoming	4	50	50

Perlu diperhatikan bahwa nilai pada jumlah kandidat adalah total kandidat yang ada pada saat pemilihan berlangsung. Setiap data uji pada tabel tersebut merepresentasikan satu peserta dengan pilihannya.

5.5 Uji Coba Kinerja Sistem

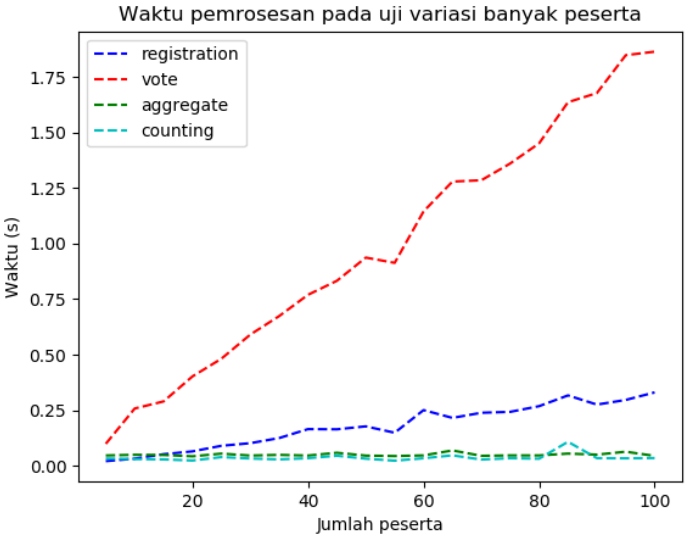
Uji coba ini dilakukan secara lokal dengan menambahkan variasi yang berupa:

1. Jumlah Peserta
   Uji coba dengan memakai variasi jumlah peserta dari 5 hingga 100 pada *state* Alabama. Uji coba ini dilakukan dengan paradigma bahwa semakin banyak data yang dimasukkan, maka beban kerja sistem akan bertambah.

2. Jumlah Kandidat
   Uji coba dengan memakai variasi jumlah kandidat dalam setiap pemungutan suara. Banyaknya kandidat disesuaikan dengan banyaknya kandidat terpilih setiap *state* sesuai dengan *dataset*. Uji coba ini dilakukan dengan paradigma bahwa semakin banyak kandidat, maka akan menambah biaya komputasi pada operasi matriksnya, dikarenakan kandidat direpresentasikan dalam bentuk matriks.

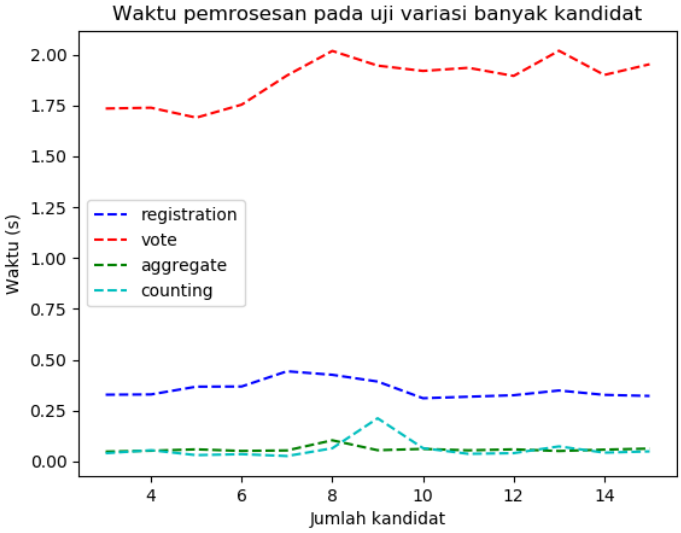
Pada setiap pengujian kinerja sistem, waktu respon program akan diukur sebagai alat ukur kinerja pada setiap tahapnya. Perlu diperhatikan bahwa waktu yang diukur merupakan waktu pemrosesan data pada sistem, sehingga tidak melibatkan waktu pada pembuatan kunci, pembuatan data uji, dll.

Hasil dari pengujian dengan variasi jumlah peserta dapat dilihat pada Gambar 5.1.



Gambar 5.1 Grafik kinerja sistem dengan menetapkan variasi jumlah peserta

Berdasarkan Gambar 5.1, waktu pemrosesan sistem untuk melakukan pemilihan adalah linier dengan penambahan yang relatif lebih tinggi daripada proses lainnya. Sedangkan untuk proses sistem melakukan agregasi dan penghitungan adalah mendekati konstan.



Gambar 5.2 Grafik kinerja sistem dengan menetapkan variasi jumlah kandidat

Berdasarkan Gambar 5.2, waktu pemrosesan sistem untuk melakukan semua proses mendekati konstan, dengan rincian waktu untuk melakukan pemilihan relatif lebih tinggi daripada waktu pemrosesan lainnya.

5.6 Analisis dan Evaluasi

Berdasarkan tabel hasil uji kebenaran pada Tabel 5.1-Tabel 5.2, dapat disimpulkan bahwa pengujian pada sistem dengan jumlah data uji sebanyak 50 data pada setiap *state* dengan jumlah kandidat yang berbeda menghasilkan data benar sebanyak 50 data. Dengan kata lain, implementasi yang diujikan berhasil memproses data masukan pemungutan suara, dan mendapat hasil sesuai dengan data awalnya.

Pengujian kinerja yang dilakukan menetapkan dua variasi, yaitu dengan variasi jumlah peserta dan jumlah kandidat. Berdasarkan hasil yang didapatkan pada Gambar 5.1, didapatkan

kesimpulan bahwa kinerja sistem pada saat melakukan penghitungan dan agregasi berjalan hampir konstan. Proses registrasi dan pemungutan suara berjalan secara linier, yang membuktikan bahwa semakin banyak data yang dimasukkan, semakin tinggi kinerja sistem. Proses yang berjalan linier tersebut disebabkan karena *voter* melakukan registrasi dan pemilihan secara seri/sekuensial.

Uji coba kinerja sistem dengan menggunakan variasi banyak kandidat yang tertera pada Gambar 5.2 menunjukkan bahwa banyaknya kandidat tidak dapat dikatakan menjadi faktor penambahan beban sistem, yang terbukti pada grafik bahwa semua waktu pemrosesan yang diukur adalah mendekati konstan.

Dari kedua uji coba kinerja yang dilakukan, dapat disimpulkan bahwa implementasi protokol MVP berjalan secara efisien dan memiliki biaya komputasi yang linier.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan terhadap implementasi dari sistem pemungutan suara elektronik pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

Dalam pengerjaan tugas akhir ini setelah melalui tahap perancangan, implementasi, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. *Model Voting Protocol* (MVP) merupakan protokol pemungutan suara elektronik yang berhasil mempertahankan kebenaran data. Hal ini telah dibuktikan dengan pengujian pada implementasi yang telah dibuat.
2. Metode *matrix masking* pada protokol MVP yang berupa *dual random matrix masking* dapat menambahkan faktor derau pada surat suara. Penambahan derau mengakibatkan hanya pihak penyelenggara saja yang dapat mengetahui data asli dari proses pemungutan suara yang berlangsung.
3. Implementasi dari sistem pemungutan suara elektronik berbasis protokol MVP dan dibuat menggunakan bahasa pemrograman Python 3 dan R dapat berjalan dengan kinerja yang baik dan efisien.

6.2 Saran

Saran yang diberikan untuk sistem pemilihan suara elektronik berbasis protokol MVP adalah dengan menggunakan jenis saluran komunikasi atau protokol pengiriman data lainnya (HTTP, FTP, UDP) pada implementasi sistem, beserta metode enkripsi pendukung lainnya.

DAFTAR PUSTAKA

[1] D. J. Bhuyan, "Effectiveness of Electronic Voting Machine in the Electoral System of India: New Opportunities and Challenges," *International Journal of Recent Technology and Engineering (IJRTE),* vol. 8, no. 2, pp. 192-199, 2019.

[2] J. L. a. D. W. S. Chow, "Robust Receipt-Free Election System with Ballot Secrecy and Verifiability," *Proc. of NDSS,* vol. 8, pp. 81-94, 2008.

[3] A. Schmidt, L. Langer, J. Buchmann dan M. Volkamer, "Specification of a Voting Service Provider," dalam *17th IEEE International Requirements Engineering Conference 2009,* 2009.

[4] O. Cetinkaya, "Analysis of Security Requirements for Cryptographic Voting Protocols (Extended Abstract)," dalam *The Third International Conference on Availability, Reliability and Security,* 2008.

[5] Y. Zhou, Y. Zhou, S. Chen dan S. S. Wu, "MVP: An Efficient Anonymous E-Voting Protocol," dalam *2016 IEEE Global Communications Conference (GLOBECOM),* Washington, DC, 2016 .

[6] S. F. M. T. Daniel Ting, "ROMM Methodology for Microdata Release," dalam *Joint UNECE/Eurostat work session on statistical data confidentiality*, Geneva, Switzerland, 2005.

[7] T. M. Neetha Francis, "An Analysis of Hybrid Cryptographic Approaches for Information Security," *International Journal of Applied Engineering Research,* vol. XIII, 2018.

[8] W. Stallings, Cryptography and Network Security: Principles and Practice, Sixth Edition, New Jersey: Pearson, 2014.

[9] “About Python,” Python, [Online]. Available: https://www.python.org/about/. [Diakses 20 Februari 2020].

[10] “R: The R Project for Statistical Computing,” [Online]. Available: https://www.r-project.org/. [Diakses 30 Mei 2020].

[11] “NumPy,” NumPy, [Online]. Available: http://www.numpy.org/. [Diakses 20 Februari 2020].

[12] “pandas,” [Online]. Available: https://pandas.pydata.org/. [Diakses 20 Februari 2020].

[13] “PyCrypto - The Python Cryptography Toolkit,” [Online]. Available: https://pycrypto.org/. [Diakses 20 Februari 2020].

[14] “Sci-Py.org,” Sci-Py.org, [Online]. Available: https://www.scipy.org/about.html. [Diakses 20 Februari 2020].

[15] “rpy2,” [Online]. Available: https://rpy2.github.io/. [Diakses 30 Mei 2020].

[16] “CRAN - Package RegSDC,” [Online]. Available: https://cran.r-project.org/web/packages/RegSDC/index.html. [Diakses 30 Mei 2020].

LAMPIRAN A: Hasil Uji Kebenaran

Tabel A.1 Hasil pengujian kebenaran pada 50 sampel data (bag. 1)

Nama *state*	Kandidat									
	1	2	3	4	5	6	7	8	9	10
Alabama	20	12	5	4	4	4	1	-	-	-
Alaska	19	16	6	4	3	1	1	-	-	-
Arizona	16	16	8	7	3	-	-	-	-	-
Arkansas	16	12	8	8	4	2	-	-	-	-
California	24	10	10	5	1	-	-	-	-	-
Colorado	27	23	-	-	-	-	-	-	-	-
Connecticut	16	15	13	4	2	-	-	-	-	-
Delaware	18	15	9	5	3	-	-	-	-	-
Florida	13	13	9	8	6	1	-	-	-	-
Georgia	12	10	10	7	7	3	1	-	-	-
Hawaii	23	9	7	7	3	1	-	-	-	-
Idaho	24	12	10	3	1	-	-	-	-	-
Illinois	18	14	13	3	1	1	-	-	-	-
Indiana	18	13	9	6	4	-	-	-	-	-
Iowa	11	9	9	8	7	2	1	1	1	1
Kansas	19	14	6	5	3	3	-	-	-	-
Kentucky	19	11	8	6	5	1	-	-	-	-
Louisiana	20	10	8	8	2	2	-	-	-	-
Maine	34	16	-	-	-	-	-	-	-	-
Maryland	17	13	8	7	5	-	-	-	-	-
Massachusetts	23	16	3	3	3	2	-	-	-	-
Michigan	19	11	9	7	3	1	-	-	-	-
Mississippi	20	16	7	4	2	1	-	-	-	-
Missouri	13	12	10	10	4	1	-	-	-	-
Montana	22	13	11	3	1	-	-	-	-	-
Nebraska	30	9	7	3	1	-	-	-	-	-
Nevada	14	11	9	9	3	3	1	-	-	-

Tabel A.2 Hasil pengujian kebenaran pada 50 sampel data (bag. 2)

New Hampshire	17	9	9	7	4	2	1	1	-	-
New Jersey	23	13	12	1	1	-	-	-	-	-
New Mexico	17	15	13	3	2	-	-	-	-	-
New York	19	16	10	4	1	1	-	-	-	-
North Carolina	14	12	12	9	2	-	-	-	-	-
North Dakota	35	15	-	-	-	-	-	-	-	-
Ohio	16	14	12	5	2	1	-	-	-	-
Oklahoma	12	11	10	8	6	3	-	-	-	-
Oregon	15	14	12	6	3	-	-	-	-	-
Pennsylvania	15	14	8	8	5	-	-	-	-	-
Rhode Island	21	15	8	4	2	-	-	-	-	-
South Carolina	16	9	7	5	4	4	3	2	-	-
South Dakota	17	15	11	4	3	-	-	-	-	-
Tennessee	20	9	7	7	3	2	2	-	-	-
Texas	13	11	9	9	5	2	1	-	-	-
Utah	28	13	6	3	-	-	-	-	-	-
Vermont	29	6	5	5	4	1	-	-	-	-
Virginia	16	10	9	7	6	1	1	-	-	-
Washington	38	4	4	3	1	-	-	-	-	-
West Virginia	19	14	9	4	4	-	-	-	-	-
Wisconsin	19	9	9	8	5	-	-	-	-	-
Wyoming	31	9	8	2	-	-	-	-	-	-

BIODATA PENULIS

Vinsensius Indra Suryanto, lahir di Kudus pada tanggal 21 Januari 1998. Penulis menempuh pendidikan mulai dari TK Cahaya Nur (2004 – 2006), SD Cahaya Nur (2006 – 2012), SMP Negeri 1 Kudus (2012 – 2014), SMA Negeri 1 Kudus (2014 – 2016). Setelah lulus dari SMAN tahun 2016, penulis mengikuti SNMPTN dan diterima di pendidikan S1 Informatika ITS pada tahun 2016 dan terdaftar dengan NRP. 05111640000064.

Di jurusan Informatika, Penulis mengambil bidang studi Algoritma dan Pemrograman. Penulis sempat aktif di dalam beberapa kegiatan kemahasiswaan, diantaranya adalah staf, staf ahli, dan pelaksana tugas kepala departemen Kesejahteraan Mahasiswa HMTC ITS, staf dan staf ahli biro keamanan dan perizinan Schematics ITS.

Komunikasi dengan penulis dapat melalui: +62-878-3156-8586 dan *email*: **vinsensiusindra@gmail.com**.