Skema Pemilu Kriptografis dengan Shamir Secret Sharing

Muhammad Rafi Ramadhan - 18223094[1,2]
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: [1]rafihalliday@gmail.com , [2]18223094@std.stei.itb.ac.id

***Abstrak*—Pemilihan umum adalah salah satu mekanisme utama untuk keberjalanan demokrasi. Di Indonesia, mekanisme yang dipakai masih manual dan melanggar berbagai prinsip seperti kerahasiaan dan kejujuran. Kedua masalah ini bisa diselesaikan dengan menggunakan metode kriptografi. Makalah ini mengeksplorasi dan mengimplementasi sebuah konsep pemilihan umum kriptografis hanya dengan menggunakan Shamir Secret Sharing sebagai mekanik perhitungan suara, bukan hanya sebagai metode enkripsi. Skema yang diajukan memakai ambang batas k dalam Shamir Secret Sharing sebagai analog untuk jumlah suara yang didapatkan calon.**

Kata kunci—Pemilihan umum, kriptografi, secret sharing, rahasia, jujur.

I. Latar Belakang & Tujuan

Pemilihan umum merupakan salah satu mekanisme utama dalam sebuah demokrasi. Di Indonesia, pemilihan umum didasarkan pada prinsip luber (langsung, umum, bebas, dan rahasia) dan jurdil (jujur dan adil). Prinsip-prinsip tersebut dicantumkan pada UU No 3. Tahun 1999. Saat ini, terdapat berbagai masalah yang menghambat pelaksanaan prinsip tersebut. Secara umum, masalah terbagi ke dalam dua kategori:

1. Kecurangan/masalah pada mekanisme pembagian, pemungutan, atau perhitungan suara:
    a. Surat suara sudah dicoblos
    b. Surat suara kurang, hilang, atau rusak
    c. Suara salah dihitung
    d. Perhitungan suara dimanipulasi panitia
2. Kebocoran/manipulasi identitas pemilih:
    a. Pemalsuan data pemilih sehingga bisa terjadi pemberian suara ganda
    b. Intimidasi terhadap pemilih

Kedua masalah ini terjadi karena mekanisme pemungutan dan perhitungan suara dilakukan secara manual. Makalah ini bertujuan memberikan sebuah skema pemilu kriptografis berlandaskan Shamir Secret Sharing yang menjawab kedua permasalahan tersebut.

II. Landasan Teori

A. Landasan Hukum

Berdasarkan UU No.3 Tahun 1999, prinsip-prinsip yang harus diikuti dalam sebuah pemilihan umum ialah:

1. Langsung:
   Pemilih berhak memberikan suara secara langsung tanpa perantara.
2. Umum:
   Semua WNI yang telah memenuhi syarat tertentu berhak memilih/dipilih.
3. Bebas:
   Setiap pemilih bebas menentukan pilihannya.
4. Rahasia:
   Pilihan pemilih dijamin tidak akan diketahui oleh pihak manapun dengan cara apapun.
5. Jujur:
   Semua pihak yang terlibat harus bertindak jujur sesuai dengan peraturan perundangan yang berlaku.
6. Adil:
   Setiap pemilih dan calon mendapat perlakuan yang sama, serta bebas dari kecurangan pihak manapun.

B. Shamir Secret Sharing

Shamir Secret Sharing adalah metode pembagian rahasia yang memungkinkan sebuah informasi dipecah menjadi beberapa bagian, kemudian dibentuk lagi tanpa perlu memakai semua pecahannya. Secara intuitif, SSS bisa dianggap seperti memecah sebuah gambar/teks menjadi beberapa bagian, kemudian menggabungkan setidaknya sebagian dari pecahan itu lalu menebak apa informasi aslinya. SSS melakukan ini secara sempurna dengan merepresentasikan informasi sebagai bilangan dan memecahnya dengan beberapa trik matematis. Metode ini

dipakai jika informasi penting harus dibagikan kepada beberapa pihak yang harus dipaksa bekerjasama.

Apabila kita ingin memecah sebuah informasi secara naif, kita bisa saja langsung memotongnya. Semisal ada password 1234, kita bisa memberikannya kepada 4 orang. Orang ke-1 mendapat angka 1, orang ke-2 mendapat angka 2, dan seterusnya. Masalahnya skema ini memaksa semua orang untuk menggabungkan potongan mereka, jika tidak lengkap informasinya hilang sama sekali. Artinya skema ini hanya cocok jika informasi hanya boleh didapatkan secara aklamasi (seperti skema 3 kunci pada kapal selam nuklir Uni Soviet), tidak cocok untuk banyak kebutuhan sehari-hari.

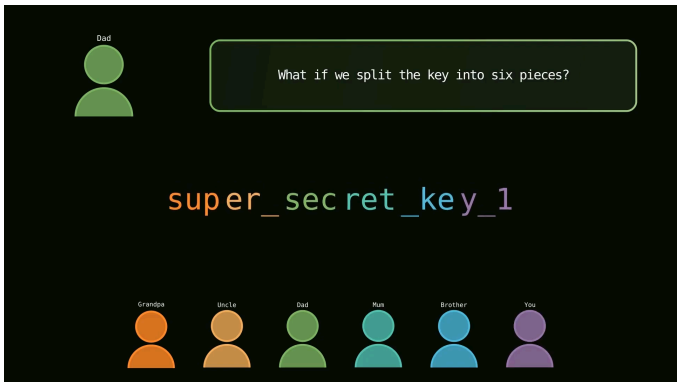


Fig. 1. Ilustrasi pembagian kunci secara naif (sumber: emcapsulation di YouTube)

Kebutuhan yang lebih sering muncul adalah “mampu mendapatkan informasi kembali dengan memakai sebagian potongan informasi, tanpa peduli potongan mana yang dipakai. Mari kita sebut “k” buah potongan dari “n” buah total potongan. Ada masalah sederhana dalam matematika yang bisa dipakai. Menebak persamaan garis dari 2 buah titik. Apabila kita mengetahui setidaknya 2 titik yang terletak pada suatu garis, kita bisa dengan mudah membuat persamaan garis itu. Seandainya hanya ada 1 saja titik yang diketahui, ada tak hingga garis yang bisa melewati titik itu.

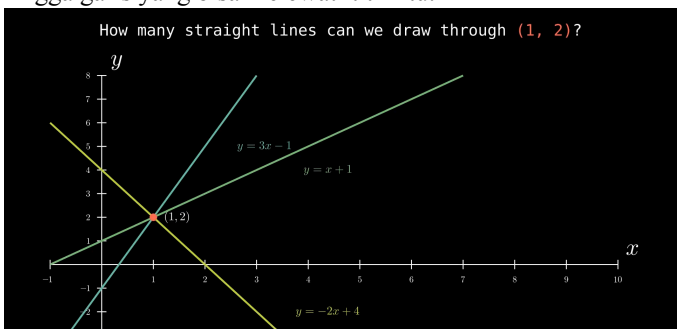


Fig. 2. Menebak persamaan garis hanya dengan 1 titik (sumber: emcapsulation di YouTube)

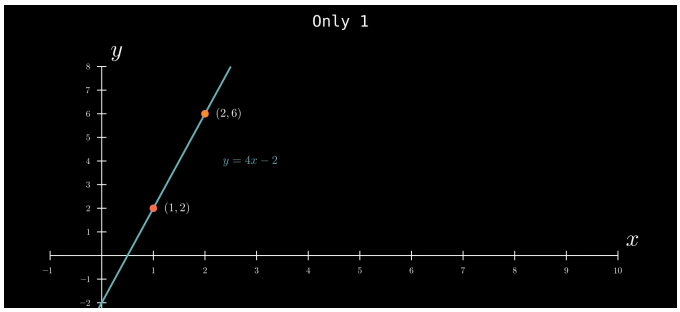


Fig. 3. Menebak persamaan garis dengan 2 titik (sumber: emcapsulation di YouTube)

Persoalan menebak garis dari 2 titik bisa diperluas. Garis adalah polinom berderajat 1, dibutuhkan 2 titik untuk menebak garis yang benar. Hal serupa berlaku pada kurva kuadrat. Kalau kita memiliki 2 titik kemudian disuruh menebak mana kurva kuadrat yang melewati 2 titik itu, jawabannya ada tak hingga. Tetapi begitu kita punya 3 titik, hanya ada 1 jawaban yang benar. Polanya berlaku terus, sebuah polinom berderajat “x” hanya bisa ditebak apabila kita punya x+1 titik.

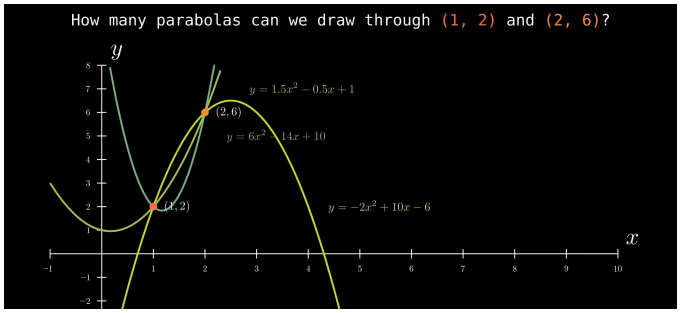


Fig. 4. Menebak persamaan parabola hanya dengan 2 titik (sumber: emcapsulation di YouTube)

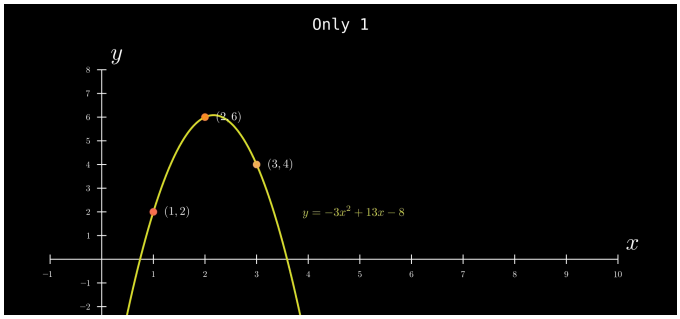


Fig. 5. Menebak persamaan parabola dengan 3 titik (sumber: emcapsulation di YouTube)

Hal ini memberikan solusi ideal untuk skema pembagian kunci kita. Kunci harus bisa didapat dari menebak polinom yang benar, metode paling sangkil adalah menganggap perpotongan kurva dengan sumbu Y (atau hasil polinom jika x-nya 0) sebagai kunci. Jadi urutannya adalah

1. Buat suatu polinom yang sesuai dengan batas minimum jumlah orang yang bekerjasama (k orang)
2. Perpotongan dengan sumbu Y dianggap kunci
3. Ambil sekian titik acak pada polinom (sesuaikan dengan jumlah total orang yang akan diberikan potongannya)
4. Saat ingin memakai kunci, kumpulkan k orang lalu cari polinom dan perpotongan sumbu Y nya

Seandainya kita memiliki 6 orang penerima potongan dan kita ingin setidaknya 4 orang bekerjasama (artinya x+1 = 4), maka kita memilih polinom derajat 3. Apabila ingin memaksa semua orang bekerjasama, cukup pertinggi saja derajat polinomnya. Sekarang kita memiliki metode yang cocok untuk banyak situasi.

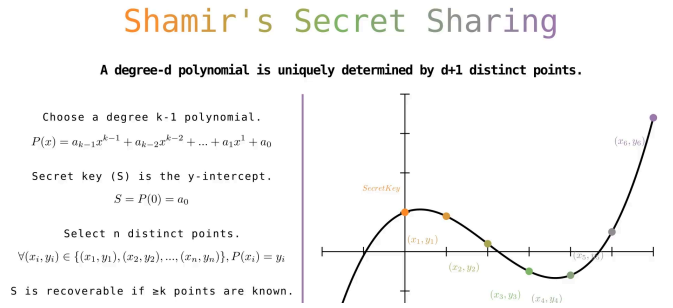


Fig. 6. Skema SSS secara umum (sumber: emcapsulation di YouTube)

Secara teori, metode tadi sudah sangat aman. Apabila kita memilih titik yang dipakai sebagai potongan secara acak, penyerang tidak akan mendapatkan informasi apapun tentang kunci asalkan mereka belum mengumpulkan cukup kunci. Artinya secara teori informasinya aman. Secara praktik, metode ini bermasalah apabila dipakai pada sistem yang tidak bisa merepresentasikan seluruh bilangan real secara akurat. Komputer merupakan sebuah sistem demikian, jumlah bilangan setelah tanda desimal terbatas, kalau kita memaksa menyimpan angka banyak-banyak, hasilnya tidak akan akurat.

```c
#include <stdio.h>

int main() {
    double pi = 3.1415926535897932384;
    printf("The value of pi is %.20lf\n", pi);
    return 0;
}
```

```
The value of pi is 3.14159265358979311600
```

Fig. 7. keterbatasan akurasi angka real pada komputer, angka yang dituliskan dalam kode berbeda dari hasil yang dicetak (sumber: penulis)

Komputer selalu akurat dengan bilangan bulat (integer), asalkan bilangannya tidak terlalu besar (sehingga overflow). Maka bisa saja kita memutuskan untuk membatasi titik yang dipakai sebagai potongan agar berupa bilangan bulat saja. Hal ini memang mempermudah implementasi dan menjamin pembentukan ulang kunci. Tetapi, sekarang penyerang tidak perlu mengecek semua bilangan real, cukup mengecek bilangan bulat saja. Akibatnya menebak polinom yang benar (berikut kuncinya) secara brute-force menjadi mudah pula.

Masalah pemakaian bilangan bulat muncul karena jumlah titik calon dipotong secara naif. Jadi, solusinya adalah memotong dengan lebih pintar, memakai operasi yang hasilnya sulit diprediksi. Operasi pembulatan ke atas dan ke bawah masih menghasilkan masalah yang sama, operasi yang lebih cocok adalah modulo. Dengan operasi modulo, 2 hal akan terjadi:

1. Persamaan direduksi ke dalam rentang [0-angka modulo], sehingga persamaan bisa dibuat memiliki derajat sangat besar tanpa khawatir menghasilkan overflow. Artinya kita bisa mengakomodasi banyak penerima potongan sekaligus.
2. Bentuk persamaan "dirusak" sehingga tidak bisa lagi ditebak dengan mudah. Persamaan polinom biasa mudah ditebak karena kontinu, operasi modulo merusak sifat ini.

Pemberian modulo pada seluruh operasi matematika yang dilakukan berarti operasi dilakukan di dalam "Medan Galois".

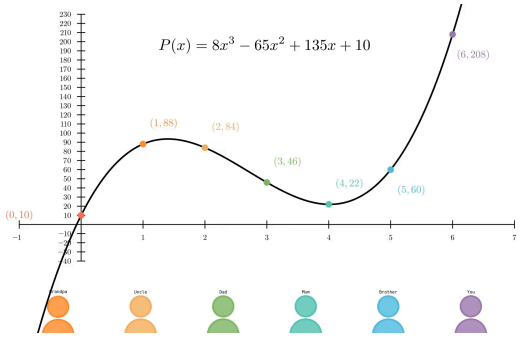


Fig. 8. Polinomial biasa (sumber: emcapsulation di YouTube)

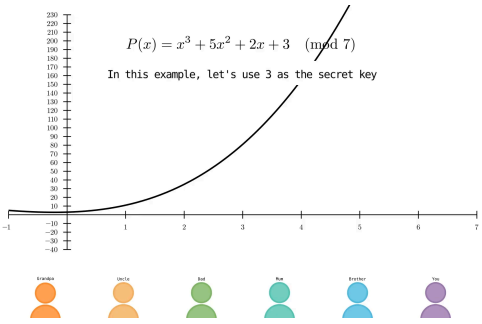


Fig. 9. Persamaan polinom setelah direduksi memakai modulo (sumber: emcapsulation di YouTube)

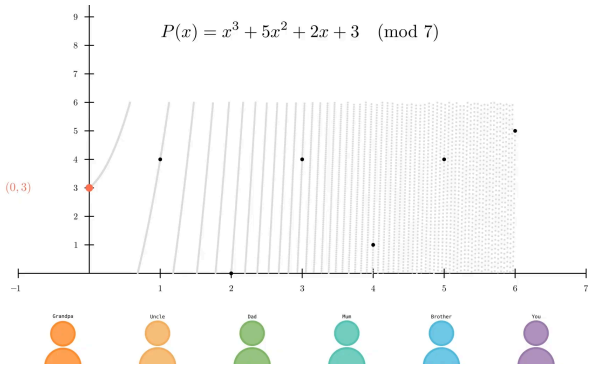


Fig. 10. Kurva hasil modulo dan titik-titik dengan x-nya berupa bilangan bulat (sumber: emcapsulation di YouTube)

Skema SSS yang dihasilkan di atas sudah aman secara teori dan praktik. Tetapi, skema ini masih punya masalah, tidak nyaman dipakai. Apabila polinomnya berderajat besar (karena jumlah penerima potongannya banyak), perhitungan untuk mencari polinom akan sangat lama. Untungnya ada trik elegan untuk masalah ini. Interpolasi Langrange. Apabila kita ingin merekonstruksi sebuah polinom, kita ingin mencari sebuah persamaan yang apabila dipakai pada semua titik tersebut.

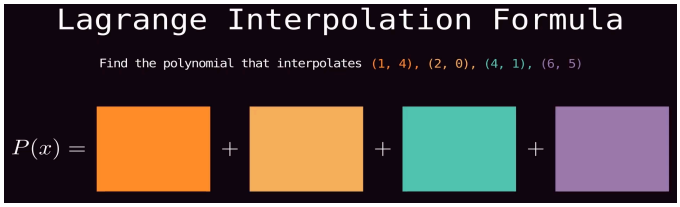


Fig. 11. Ilustrasi polinom sebagai fungsi dengan penanda "aktif"/"mati" (sumber: emcapsulation di YouTube)

Semisal ada polinom P(x) yang melewati 4 titik (1,4), (2,0), (4,1), (6,5). Artinya:

- $1,4 \rightarrow P(1) = a(1)^3 + b(1)^2 + c(1) + d = 4$
- $2,0 \rightarrow P(2) = a(2)^3 + b(2)^2 + c(2) + d = 0$
- $4,1 \rightarrow P(4) = a(4)^3 + b(4)^2 + c(4) + d = 1$
- $6,5 \rightarrow P(1) = a(6)^3 + b(6)^2 + c(6) + d = 5$

Secara intuitif, ini sama saja dengan mencari suatu fungsi yang akan "aktif" di satu tempat saja. Anggap fungsi fiktif kita ini menyatakan:

$$P(x) = flag_1(4) + flag_2(0) + flag_3(1) + flag_4(5)$$

Setiap penanda/flag hanya akan aktif pada kondisi tertentu. $flag_1$ misalnya hanya akan menyala apabila x-nya 1. Sisanya mati dan bernilai 0. Dengan demikian didapat P(1) = 4. Perilaku demikian juga kita anggap berlaku pada semua penanda/flag lain. Artinya, penanda/flag yang kita perlukan adalah penanda yang akan bernilai 0 apabila x-nya tidak sama dengan x untuk suatu titik. Mari kita tinjau kasus P(1) = 4.

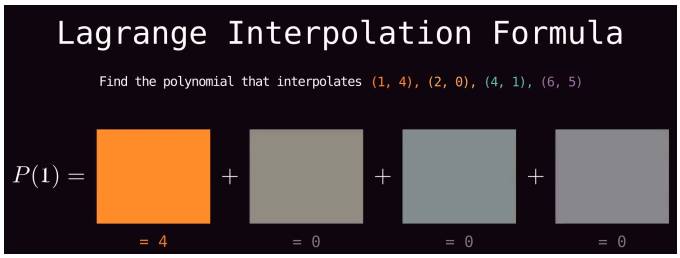


Fig. 12. Kasus aktivasi P(1) = 4 (sumber: emcapsulation di YouTube)

Untuk mendapat perilaku ini, kita bisa menambahkan (x-1) pada semua bagian polinomial kecuali bagian pertama (yang berisi 4, harus aktif). Hal serupa bisa dilakukan pada semua bagian yang lain, tambahkan $(x - x')$ dengan x' merupakan x dari semua titik yang lain.

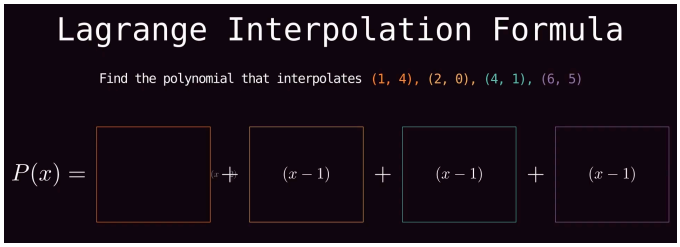


Fig. 13. Mengaktifkan bagian pertama dan mematikan sisanya (sumber: emcapsulation di YouTube)

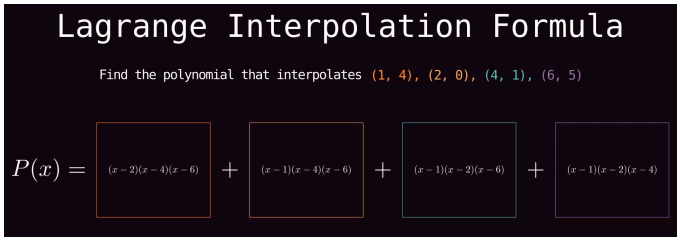


Fig. 14. Aktivasi dan deaktivasi yang lengkap (sumber: emcapsulation di YouTube)

Hasil dari interpolasi ini adalah polinomial berikut:

$$\begin{aligned} P(x) &= 1\,(x-2)(x-4)(x-6) \\ &+ 0(x-1)(x-4)(x-6) \\ &+ 1(x-1)(x-2)(x-6) \\ &+ 5(x-1)(x-2)(x-4) \end{aligned}$$

Polinomial hasil interpolasi ini sudah memberikan bagian yang diharapkan saja. Akan tetapi, bagian yang "aktif" tersebut hasilnya dirusak oleh penanda/flag. Saat x = 6, semua bagian kecuali bagian terakhir akan bernilai 0 karena memiliki suku (x-6). Sementara itu, bagian terakhir akan bernilai $5(6-1)(6-2)(6-4) = 5(5)(4)(2) = 5(40)$. Agar polinomial memberikan hasil yang benar, kita cukup mengeliminasi ketiga suku penanda/flag, bagi saja mereka dengan dirinya.

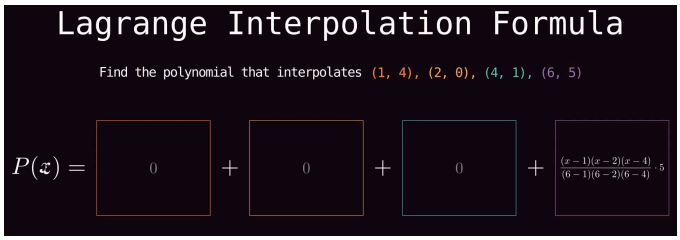


Fig. 15. Interpolasi Langrange untuk bagian terakhir (sumber: emcapsulation di YouTube)

Hal yang sama dilakukan untuk semua bagian yang lain.

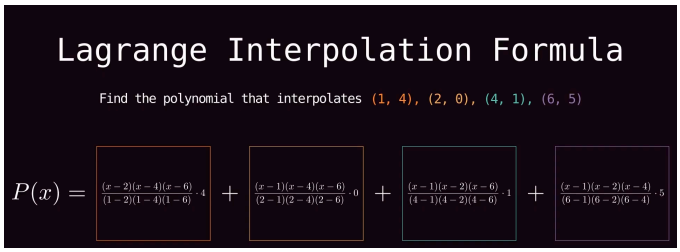


Fig. 16. Interpolasi Langrange secara lengkap (sumber: emcapsulation di YouTube)

Formula yang melakukan semua manipulasi di atas adalah:

$$P(x) = \sum_{i=1}^{k} y_i \prod_{1 \le j \le k,\, j \ne i} \frac{x - x_j}{x_i - x_j}$$

Untuk memakai formula ini, pakai semua titik yang dimiliki (minimal sejumlah batas k) untuk membentuk P(x) kemudian cari P(0). Itu adalah kunci hasil rekonstruksi. Perlu diingat bahwa polinom beroperasi dalam modulo sehingga semua operasi (termasuk pembagian) harus dilakukan dalam modulo (Medan Galois) juga.

III. Proposal Skema

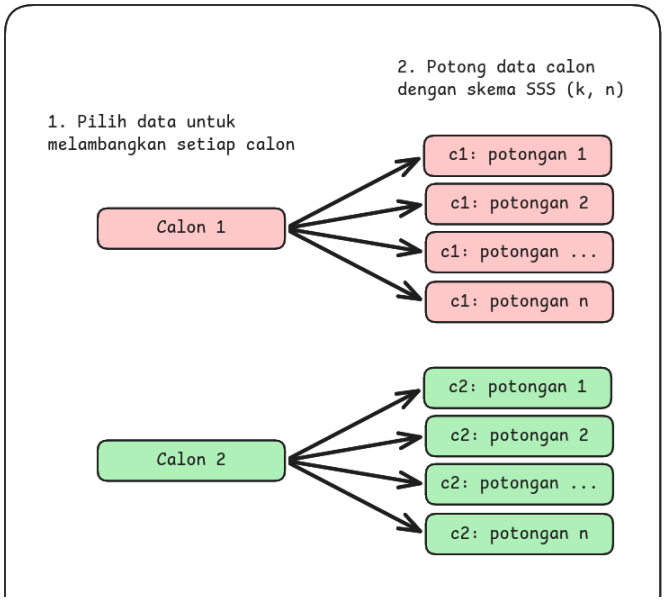
A. *Skema Naif*

Dalam pemilihan, pertanyaan yang diajukan sekaligus dijawab adalah "apakah calon mendapatkan suara dengan jumlah tertentu?" Hal ini sama persis dengan rekonstruksi sebuah data dalam Shamir Secret Sharing (SSS), "apakah jumlah potongan/share yang dikumpulkan mencukupi jumlah tertentu (ambang batas k)?" Dalam SSS, "k" adalah jumlah potongan minimal yang harus dikumpulkan untuk bisa mendapatkan kembali data yang awalnya dibagi dan "n" adalah jumlah orang yang menerima potongan data. Skema SSS (2,3) artinya data dibagi ke 3 orang dan diperlukan minimal 2 potongan untuk mendapatkan datanya kembali. Untuk menerjemahkan pemilu menjadi sebuah skema SSS, jumlah suara minimal yang harus didapatkan untuk menang pemilihan adalah "k" dan jumlah pemilih adalah "n".

Skema yang diusulkan adalah sebagai berikut:

1. Setiap calon yang bisa dipilih akan dilambangkan dengan sebuah data.
    a. Data ini bisa berupa apapun (misal nama lengkap calon)
    b. Data harus unik (nama lengkap calon bisa dipakai kalau tidak ada yang sama)
    c. Data harus bisa ditulis sebagai angka agar bisa diproses komputer (nama adalah rangkaian karakter dan bisa diproses komputer)
2. Data setiap calon dipotong menggunakan skema SSS (k, n) menjadi n buah potongan/share
    a. k adalah jumlah suara minimal yang harus diterima calon agar dinyatakan menang
    b. n adalah jumlah pemilih
3. Setiap pemilih menerima sebuah surat suara berupa potongan/share untuk setiap calon
4. Pemilih memberikan potongan/share untuk calon yang dia pilih
5. Setelah masa pemungutan suara berakhir, panitia pemilihan melakukan Interpolasi Lagrange pada setiap suara calon. Calon yang datanya berhasil disusun kembali dipastikan menerima suara sejumlah k atau lebih

Untuk memudahkan dan mengamankan pemilihan, surat suara bisa berupa sebuah QR code untuk setiap calon. Pemilih akan membawa QR calon yang dia pilih dan memindainya di kotak suara. Pada saat perhitungan suara, suara baru dipisahkan berdasarkan calon (karena memakai potongan dari polinom berbeda akan merusak hasil Interpolasi Lagrange). Dengan mekanisme ini, tidak ada yang mengetahui siapa calon yang dipilih oleh seorang pemilih dan berikut hasil pemilihan. Hasil pemilihan baru bisa diketahui di akhir.



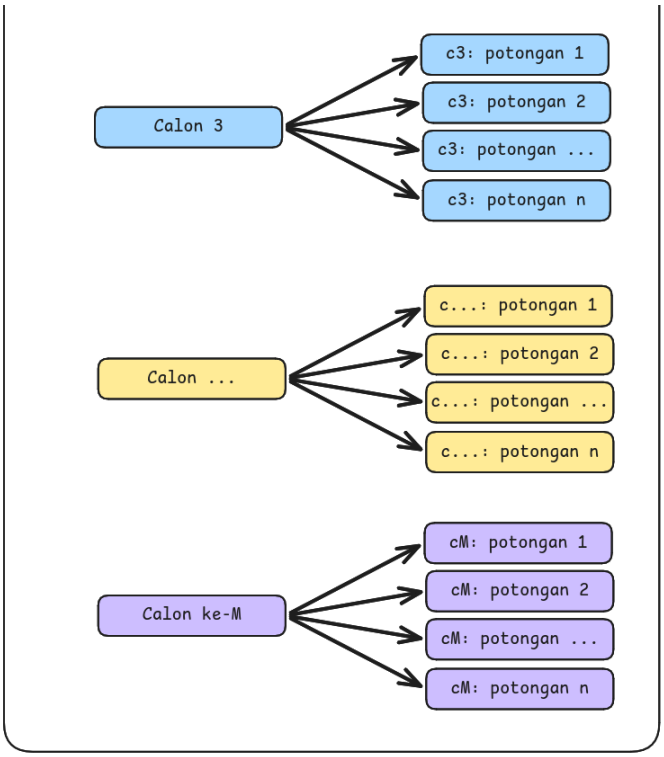


Fig. 17. Langkah 1 dan 2, pemilihan data dan pembagian data (sumber: penulis)

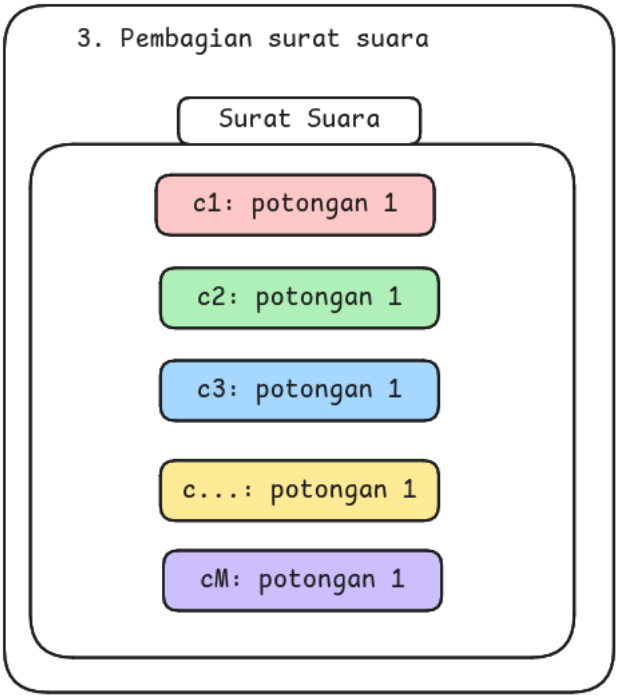


Fig. 18. Langkah 3, pembagian suara (sumber: penulis)

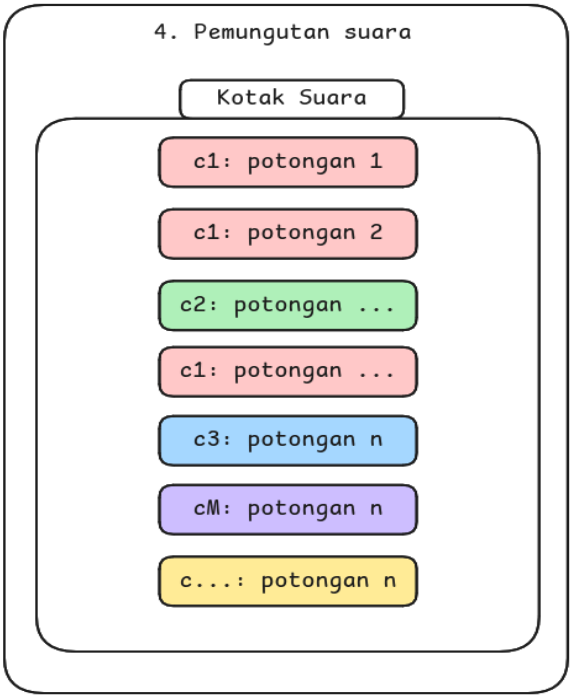


Fig. 19. Langkah 4, pemungutan suara (sumber: penulis)

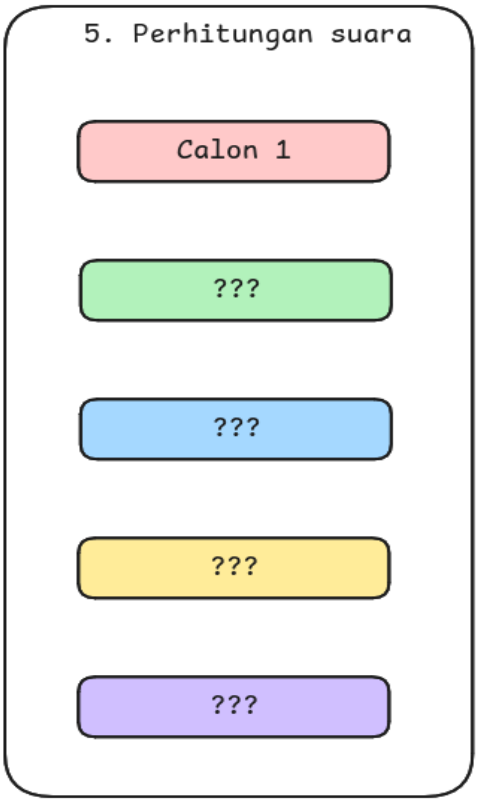


Fig. 20. Langkah 5, perhitungan suara (sumber: penulis)

Pertimbangkan contoh sederhana berikut:

- Ada 3 calon: Alice, Bob, Clara
- Data yang dipakai adalah nama lengkap calon (nama mereka berbeda semua)
- Ada 100 orang pemilih
- Kemenangan berdasarkan mayoritas sederhana (50% + 1, artinya minimal 51 suara agar menang)

Skema yang akan dipakai untuk contoh di atas adalah:

1. Setiap calon yang bisa dipilih dilambangkan dengan data
    a. Data berupa nama lengkap
    b. Karena nama lengkap unik, data boleh dipakai
    c. Nama lengkap adalah rangkaian karakter (string) dan bisa dilambangkan sebagai angka
2. Data setiap calon dipotong menggunakan skema SSS (51, 100) menjadi 100 buah potongan/share
    a. Seorang calon harus menerima minimal 51 suara agar dinyatakan menang
    b. Terdapat 100 orang pemilih
3. Semua pemilih (100 orang) menerima sebuah surat suara berupa potongan/share untuk setiap calon, artinya setiap orang menerima 3 potongan/share:
    a. 1 untuk memilih Alice
    b. 1 untuk memilih Bob
    c. 1 untuk memilih Clara
4. Pemilih memberikan potongan/share untuk calon yang dia pilih
5. Semisal perolehan suara adalah sebagai berikut:
    a. Alice: 51 suara
    b. Bob: 30 suara
    c. Clara: 19 suara
6. Data Alice yang akan terbuka, sementara data calon lainnya akan gagal dibuka

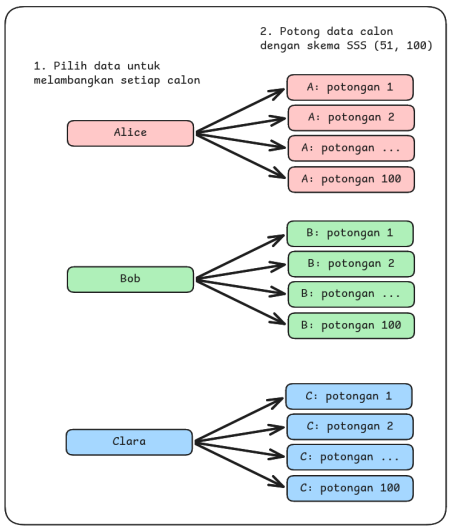


Fig. 21. Data untuk Alice, Bob, dan Clara berupa nama mereka. Data dibagi dengan SSS (51, 100) (sumber: penulis)

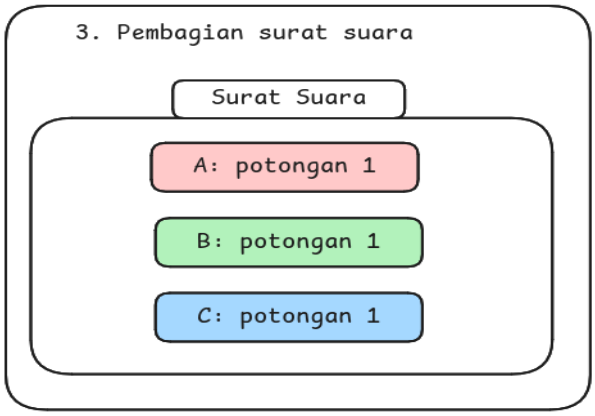


Fig. 22. Bentuk konseptual surat suara pada pemilihan ini (sumber: penulis)

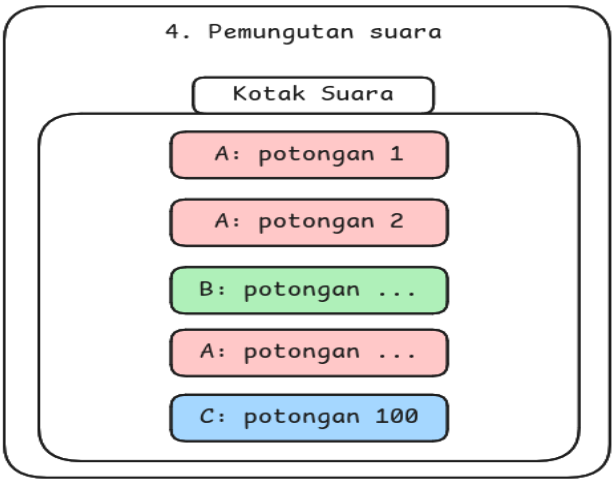


Fig. 23. Kotak suara secara konseptual (sumber: penulis)

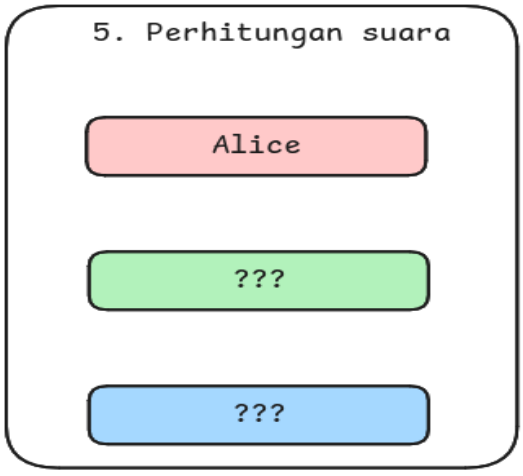


Fig. 24. Hasil perhitungan suara, data Alice berhasil dibuat kembali, Alice memenangkan pemilihan (sumber: penulis)

Skema yang diajukan di atas bisa menangani kasus dimana hasil pemilihan memberikan seorang calon setidaknya 50%+1 dari total jumlah suara. Akan tetapi, skema ini akan gagal memberikan hasil apabila tidak ada seorang pun yang menerima suara sebanyak itu. Pada kehidupan nyata, seringkali hasil yang muncul tidak memenuhi mayoritas sederhana. Contohnya, apabila ada 3 calon dalam sebuah pemilu, bisa saja hasil yang diperoleh adalah 40:35:25, pada kasus ini, skema mayoritas sederhana gagal. Untuk menangani masalah ini, diperlukan skema yang lebih baik.

B. Skema Hitung Kasar

Untuk mengatasi permasalahan pada skema sebelumnya, diperlukan skema yang lebih kreatif. Pertanyaan utama yang ditanyakan pada saat akan merekonstruksi sebuah data yang dibagi dengan Shamir Secret Sharing adalah "apakah kita memiliki setidaknya k buah potong". Hal ini bisa secara langsung diterjemahkan ke perolehan suara sebagaimana dilakukan di skema sebelumnya (Skema Naif). Skema sebelumnya hanya menanyakan 1 buah pertanyaan, "apakah calon mendapatkan suara sebanyak X?", dengan X sebagai batas minimum perolehan suara agar seorang calon menang. Kenapa tidak menanyakan hal serupa berkali-kali?

- Apakah calon mendapatkan suara sebanyak 10%?
- Apakah calon mendapatkan suara sebanyak 20%?
- dan seterusnya

Dengan mengembangkan Skema Naif lebih jauh, suara bisa dihitung sampai akurasi tertentu dengan tingkat akurasi sesuka hati. Akurasi ini bisa dianggap sebagai granularitas, seberapa kecil pencacahan suara yang diinginkan. Entah itu per 10%, 1%, atau bahkan lebih kecil lagi. Secara teori, granularitas bisa dibuat sangat kecil (atau sangat halus) sampai kita bisa menghitung persis berapa suara yang diperoleh oleh seorang calon. Akan tetapi, karena Interpolasi Lagrange memiliki kompleksitas waktu $O(n^2)$, memakai granularitas yang sangat halus akan mengakibatkan perhitungan suara berjalan terlalu lama. Disarankan memakai granularitas yang tidak terlalu halus, seperti memotong per 1%. Skema baru ini disebut Skema Hitung Kasar.

Skema yang diusulkan adalah sebagai berikut:

1. Setiap calon yang bisa dipilih akan dilambangkan dengan sebuah data.
    a. Data ini bisa berupa apapun (misal nama lengkap calon)
    b. Data harus unik (nama lengkap calon bisa dipakai kalau tidak ada yang sama)
    c. Data harus bisa ditulis sebagai angka agar bisa diproses komputer (nama adalah rangkaian karakter dan bisa diproses komputer)
2. Pilih granularitas, seberapa halus akurasi perhitungan suara yang diinginkan (100/granularitas = akurasi, apabila ingin akurasi per 10%, gunakan granularitas = 10, apabila ingin akurasi per 1%, gunakan granularitas = 1)
3. Turunkan dan pasangkan g data untuk calon, jumlah data per calon (g) adalah granularitas. Data ini disebut data pencacah dan bisa diturunkan dengan fungsi hash.
4. Setiap data pencacah dipotong dengan aturan berikut:
    a. i dimulai dari 1
    b. Data pencacah ke-i dipotong dengan skema SSS ($(\frac{i}{g} \times n)$, n)
    c. k adalah jumlah suara minimal yang harus diterima agar data pencacah ke-i bisa dibuka
    d. n adalah jumlah pemilih
    e. g adalah granularitas
5. Setiap pemilih menerima sebuah surat suara berupa seluruh potongan/share untuk setiap calon
6. Pemilih memberikan potongan/share untuk calon yang dia pilih
7. Setelah masa pemungutan suara berakhir, panitia pemilihan melakukan Interpolasi Lagrange untuk menentukan perolehan suara dengan melakukan iterasi terhadap semua data pencacah:
    a. i dimulai dari 1
    b. data pencacah ke-i akan direkonstruksi
    c. apabila rekonstruksi gagal, rekonstruksi berikutnya juga akan gagal, berhenti
8. Akibatmya, diketahui bahwa seorang calon mendapatkan suara setidaknya sebanyak $i \times (\frac{n}{g})$

Terdapat perubahan pada langkah pemasangan data. Pada Skema Naif, setiap calon hanya memiliki 1 data yang harus dia rekonstruksi/buka. Pada skema hasil pengembangan, setiap calon akan diberikan g buah data. Artinya, sekarang pemilih akan menerima g buah potongan/share untuk setiap calon. Proses perhitungan suara juga berubah, sekarang dilakukan perhitungan sebanyak g kali karena ada data pencacah sebanyak g. Suara yang diperoleh seorang calon setidaknya sama dengan suara yang diperlukan untuk membuka data pencacah terakhir yang berhasil dibuka seorang calon.

Pertimbangkan contoh berikut:

- Ada 3 calon: Alice, Bob, Clara
- Data yang dipakai adalah nama lengkap calon (nama mereka berbeda semua)
- pemilih (n) = 100
- granularitas (g) = 10
- Alice mendapat 50 suara
- Bob mendapat 40 suara
- Clara mendapat 10 suara

Skema yang akan dipakai untuk contoh di atas adalah:

1. Setiap calon yang bisa dipilih akan dilambangkan dengan sebuah data.
    a. Data ini bisa berupa apapun (misal nama lengkap calon)
    b. Data harus unik (nama lengkap calon bisa dipakai kalau tidak ada yang sama)

    c. Data harus bisa ditulis sebagai angka agar bisa diproses komputer (nama adalah rangkaian karakter dan bisa diproses komputer)
2. Granularitasnya 10
3. Setiap calon mendapat 10 pencacah
4. Setiap data pencacah dipotong dengan aturan berikut:
    a. i antara 1-10
    b. Setiap data pencacah dipotong dengan skema SSS (10 * i, 100)
5. Setiap pemilih menerima sebuah surat suara berupa seluruh potongan/share untuk setiap calon
6. Pemilih memberikan potongan/share untuk calon yang dia pilih
7. Setelah masa pemungutan suara berakhir, panitia pemilihan melakukan Interpolasi Lagrange untuk menentukan perolehan suara dengan melakukan iterasi terhadap semua data pencacah:
    a. i antara 1-10
    b. Setiap data pencacah dibuka dengan skema SSS (10 * i, 100)
8. Hasil pemilu adalah sebagai berikut:
    a. Alice mendapat 50 suara, artinya ada 5 data pencacah terbuka, diketahui setidaknya 50% suara diperolah
    b. Bob mendapat 40 suara, artinya ada 3 data pencacah terbuka, diketahui setidaknya 40% suara diperolah
    c. Clara mendapat 10 suara, artinya ada 1 data pencacah terbuka, diketahui setidaknya 10% suara diperolah

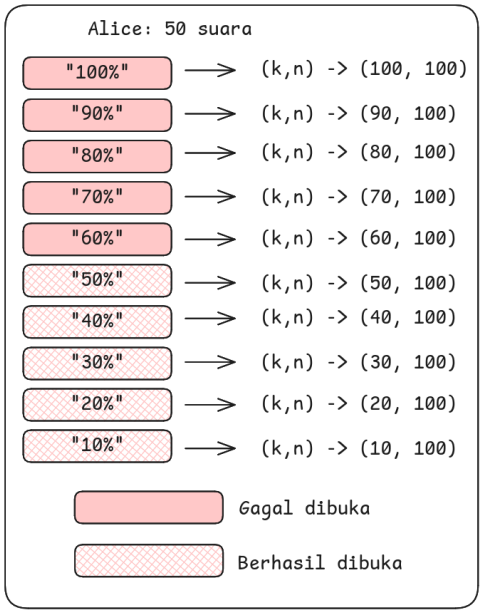


Fig. 25. Kondisi akhir data pencacah Alice (sumber: penulis)

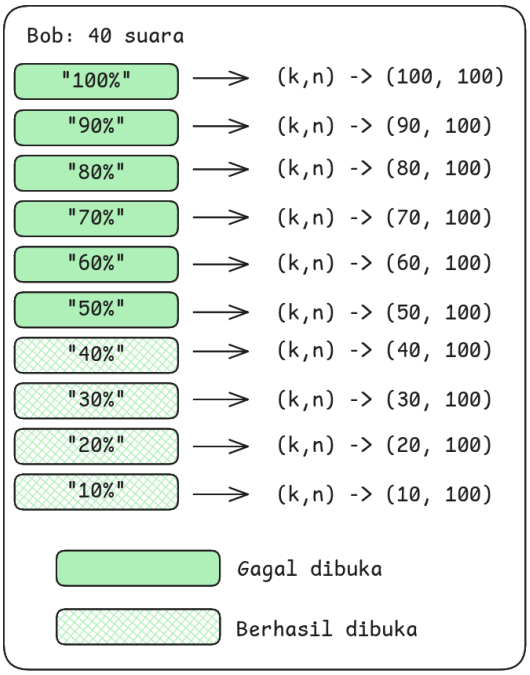


Fig. 26. Kondisi akhir data pencacah Bob (sumber: penulis)

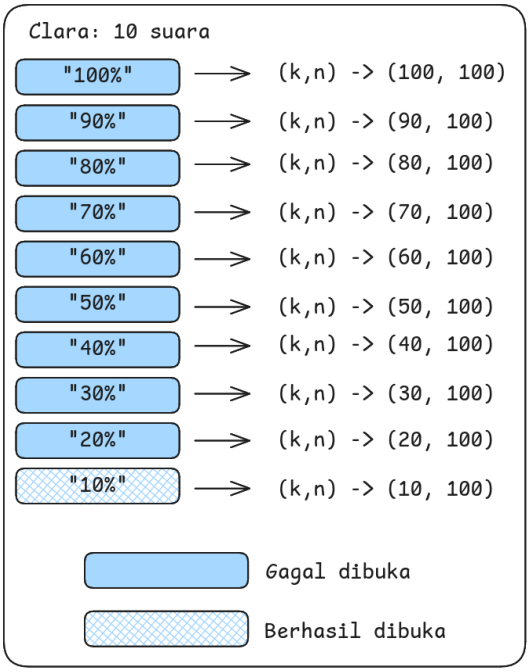


Fig. 27. Kondisi akhir data pencacah Clara (sumber: penulis)

Dengan Skema Naif, tidak ada pemenang jelas karena Alice pun dengan suara terbanyak masih akan gagal membuka kembali data miliknya. Pada skema baru ini, Alice berhasil

membuka data pencacah terbanyak. Maka Alice dinyatakan sebagai pemenang dalam pemilihan ini. Skema ini bisa diperluas sampai akurasi yang dianggap perlu dengan cara menambah granularitasnya. Dengan demikian, permasalahan pada Skema Naif berhasil diselesaikan.

Walau skema ini berhasil menyelesaikan permasalahan pada Skema Naif, skema ini akan gagal apabila selisih perolehan suara kurang dari granularitas dan perolehan suara masih berada rentang untuk membuka data pencacah yang sama. Contohnya adalah skenario pemilihan dengan 3 calon dan perolehan suara 49:48:3. Seandainya granularitas yang dipakai adalah 10, calon ke-1 dan ke-2 akan berhasil membuka 4 data pencacah. Keduanya dinyatakan "mendapat setidaknya 40% suara" tanpa bisa diketahui berapa persentase persisnya. Skema Hitung Kasar masih harus dikembangkan lebih jauh.

C. Skema Hitung Halus

Masalah pada Skema Hitung Kasar bisa diatasi dengan pengembangan lebih jauh. Apabila kita telaah pertanyaan umum yang diajukan pada saat ingin merekonstruksi suatu data pada Shamir Secret Sharing, "apakah ada cukup potongan/share tersedia?", ada jawaban sederhana. Kita bisa membuka data apabila ada cukup potongan, kalau tidak, tambahkan terus sampai datanya terbuka. Mekanik sederhana ini bisa dimanfaatkan untuk menghitung suara sampai presisi satuan. Sederhananya, gunakan Skema Hitung Kasar, tetapi apabila hasil akhir masih seri, tambahkan suara satu per satu sampai data pencacah bisa dibuka. Masalahnya, suara dari mana? Suara tidak bisa diambil sembarangan dari pemilih lain. Solusinya dengan menambah jumlah potongan di awal. Cara kerja Shamir Secret Sharing memungkinkan kita untuk membagikan potongan/share sebanyak yang kita suka.

Skema yang diusulkan adalah sebagai berikut:

1. Setiap calon yang bisa dipilih akan dilambangkan dengan sebuah data.
    a. Data ini bisa berupa apapun (misal nama lengkap calon)
    b. Data harus unik (nama lengkap calon bisa dipakai kalau tidak ada yang sama)
    c. Data harus bisa ditulis sebagai angka agar bisa diproses komputer (nama adalah rangkaian karakter dan bisa diproses komputer)
2. Pilih granularitas, seberapa halus akurasi perhitungan suara yang diinginkan (100/granularitas = akurasi, apabila ingin akurasi per 10%, gunakan granularitas = 10, apabila ingin akurasi per 1%, gunakan granularitas = 1)
3. Turunkan dan pasangkan g data untuk calon, jumlah data per calon (g) adalah granularitas. Data ini disebut data pencacah dan bisa diturunkan dengan fungsi hash.
4. Setiap data pencacah dipotong dengan aturan berikut:
    a. i dimulai dari 1
    b. Data pencacah ke-i dipotong dengan skema SSS ($(\frac{i}{g} \times n)$, 2n), setengah untuk pemilih, setengah untuk perhitungan di akhir (disebut share penghitung)
    c. k adalah jumlah suara minimal yang harus diterima agar data pencacah ke-i bisa dibuka
    d. n adalah jumlah pemilih
    e. g adalah granularitas
5. **Potongan/share dikirim ke pemiliih dan penyimpanan sementara**
    a. Setiap pemilih menerima sebuah surat suara berupa seluruh potongan/share untuk setiap calon, total surat suara yang dikirim ke pemilih adalah sebanyak n
    b. Panitia menyimpan n buah surat suara untuk kebutuhan perhitungan nanti (share penghitung)
6. Pemilih memberikan potongan/share untuk calon yang dia pilih
7. Setelah masa pemungutan suara berakhir, panitia pemilihan melakukan Interpolasi Lagrange untuk menentukan perolehan suara dengan melakukan iterasi terhadap semua data pencacah:
    a. i dimulai dari 1
    b. data pencacah ke-i akan direkonstruksi
    c. apabila rekonstruksi gagal, rekonstruksi berikutnya juga akan gagal, berhenti
8. Akibatmya, diketahui bahwa seorang calon mendapatkan suara setidaknya sebanyak $i \times (\frac{n}{g})$
9. **Apabila terjadi seri antara calon, share penghitung akan digunakan seperti berikut:**
    a. Ambil data pencacah TERENDAH yang GAGAL dibuka oleh calon yang seri
    b. Ambil share penghitung untuk data pencacah tersebut
    c. Tambahkan share penghitung satu per satu bersama dengan surat suara asli untuk mencoba membuka data pencacah tersebut
    d. Calon yang berhasil membuka data pencacah terlebih dahulu dinyatakan menang
    e. Apabila ada lebih dari 1 calon yang berhasil, berarti benar-benar seri

Perhatikan perubahan yang terjadi, sekarang jumlah surat suara yang dibangkitkan dua kali lipat dan perhitungan akhir diberikan langkah tambahan. Skema ini bisa memberikan perhitungan yang akurat sampai ke angka satuan tanpa memerlukan granularitas yang terlalu kecil.

Pertimbangkan contoh berikut:

- Ada 3 calon: Alice, Bob, Clara
- Data yang dipakai adalah nama lengkap calon (nama mereka berbeda semua)

- pemilih (n) = 100
- granularitas (g) = 10
- Alice mendapat 49 suara
- Bob mendapat 48 suara
- Clara mendapat 3 suara

Skema yang akan dipakai untuk contoh di atas adalah:

1. Setiap calon yang bisa dipilih akan dilambangkan dengan sebuah data.
    a. Data ini bisa berupa apapun (misal nama lengkap calon)
    b. Data harus unik (nama lengkap calon bisa dipakai kalau tidak ada yang sama)
    c. Data harus bisa ditulis sebagai angka agar bisa diproses komputer (nama adalah rangkaian karakter dan bisa diproses komputer)
2. Granularitas 10
3. Turunkan dan pasangkan g data untuk calon, jumlah data per calon (g) adalah granularitas. Data ini disebut data pencacah dan bisa diturunkan dengan fungsi hash.
4. Setiap data pencacah dipotong dengan aturan berikut:
    a. i antara 1-10
    b. Setiap data pencacah dipotong dengan skema SSS (10 * i, 200)
5. Potongan/share dikirim ke pemiliih dan penyimpanan sementara
    a. 100 untuk pemilih
    b. 100 disimpan sebagai share penghitung
6. Pemilih memberikan potongan/share untuk calon yang dia pilih
7. Setelah masa pemungutan suara berakhir, panitia pemilihan melakukan Interpolasi Lagrange untuk menentukan perolehan suara dengan melakukan iterasi terhadap semua data pencacah:
    a. i dimulai dari 1
    b. data pencacah ke-i akan direkonstruksi
    c. apabila rekonstruksi gagal, rekonstruksi berikutnya juga akan gagal, berhenti
8. Hasil pemilu adalah sebagai berikut:
    a. Alice berhasil membuka 4 data pencacah, artinya mendapat setidaknya 40% suara
    b. Bob berhasil membuka 4 data pencacah, artinya mendapat setidaknya 40% suara
    c. Clara tidak berhasil membuka data pencacah sama sekali, dipastikan kurang dari 10% suara
9. Karena Alice dan Bob masih seri, share penghitung dipakai
    a. Data pencacah terendah yang gagal dibuka adalah pencacah ke-5, dengan ambang batas 50
    b. Data pencacah ke-5 untuk Alice dan Bob diinterpolasi dengan suara hasil pemilu ditambah 1 share penghitung, apabila masih gagal, share penghitung ditambahkan lagi, ulangi sampai ada yang berhasil duluan
    c. Setelah penambahan 1 share penghitung, Alice berhasil membuka data pencacah ke-5 (dengan 49 suara hasil pemilu + 1 share penghitung).
    d. Bob gagal membuka data pencacah ke-5 (48 suara hasil pemilu + 1 share penghitung = 49, tidak cukup, minimal 50)

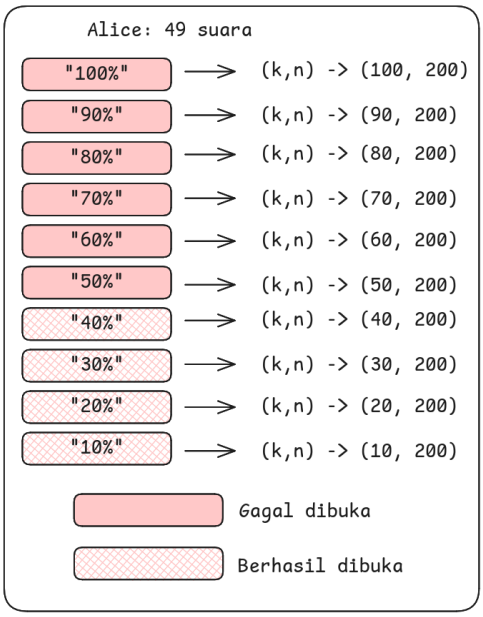


Fig. 28. Kondisi data pencacah Alice (sumber: penulis)

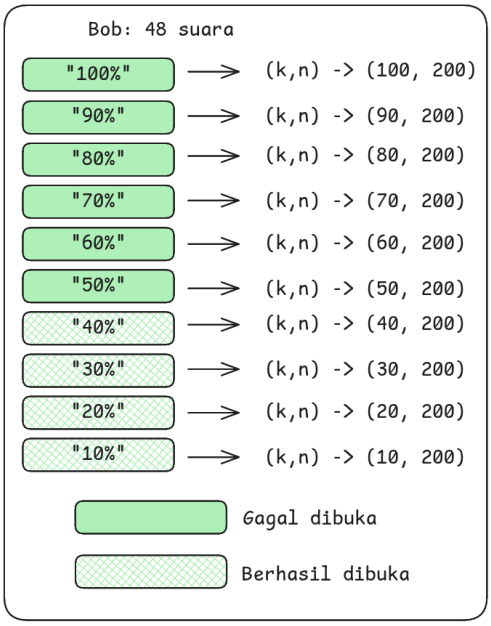


Fig. 29. Kondisi data pencacah Bob (sumber: penulis)

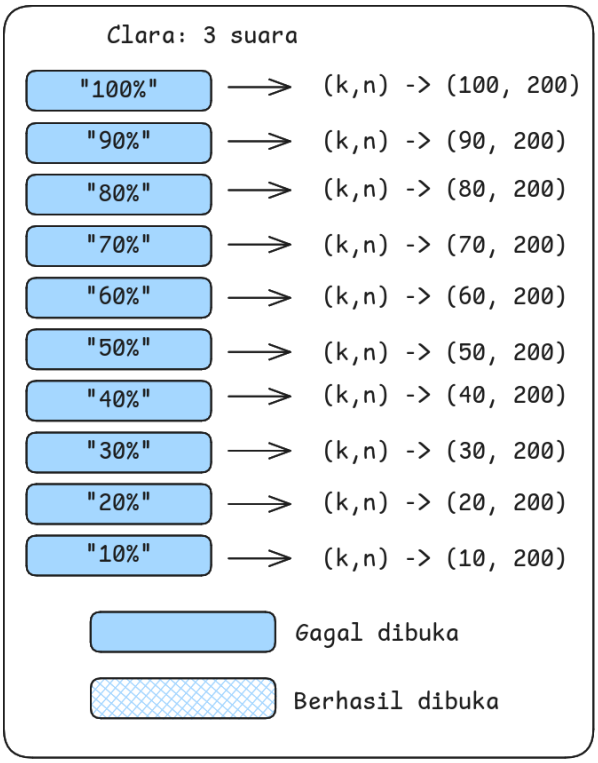


Fig. 30. Kondisi data pencacah Clara (sumber: penulis)

Setelah perhitungan awal, Alice dan Bob mengalami seri. Kebuntuan ini dipecahkan dengan memakai share penghitung.

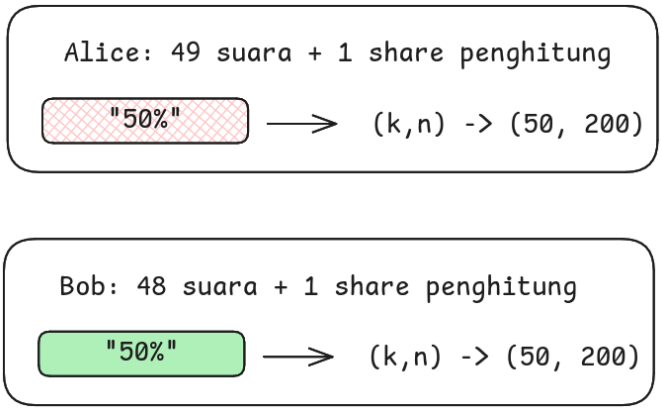


Fig. 31. Memecah kebuntuan dengan share penghitung(sumber: penulis)

IV. IMPLEMENTASI

Kode lengkap bisa dilihat di github, tautan diberikan di bagian akhir. Kode yang ditampilkan disini adalah kode penghitung suara untuk masing-masing skema.

A. Implementasi Skema Naif

```python
# Hitung suara
def hitung_suara(prime, folder="kotak_suara"):
    shares_per_calon = {}

    for filename in os.listdir(folder):
        if not filename.endswith(".json"):
            continue
        filepath = os.path.join(folder, filename)
        with open(filepath, 'r') as f:
            data = json.load(f)

        nomor = data["nomor"]
        x, y = data["x"], data["y"]

        if nomor not in shares_per_calon:
            shares_per_calon[nomor] = []
        shares_per_calon[nomor].append((x, y))

    # Rekonstruksi data disini, kalo gak bisa berarti suara dia kurang
    hasil = {}
    for nomor, shares in shares_per_calon.items():
        reconstruted = sss.reconstruct_data(shares, prime)
        try:
            nama = int_to_string(reconstruted)
            hasil[nomor] = ("menang", nama)
        except (UnicodeDecodeError, OverflowError):
            hasil[nomor] = ("gagal", None)

    return hasil
```

Fig. 32. Fungsi untuk menghitung suara dengan Skema Naif

Skema Naif hanya menangani mayoritas sederhana dalam pemilihan. Skema ini tidak bisa menentukan pemenang apabila semua calon gagal mencapai mayoritas sederhana (50%+1).

B. Implementasi Skema Hitung Kasar

```python
def hitung_suara_kasar(nama_calon_map, granularitas, prime, folder="kotak_suara"):
    shares_per_calon = {}

    for filename in os.listdir(folder):
        if not filename.endswith(".json"):
            continue
        filepath = os.path.join(folder, filename)
        with open(filepath, 'r') as f:
            data = json.load(f)

        nomor = data["nomor"]
        if nomor not in shares_per_calon:
            shares_per_calon[nomor] = {g: [] for g in range(1, granularitas+1)}

        for g in range(1, granularitas+1):
            x = data[f"data_{g}"]["x"]
            y = data[f"data_{g}"]["y"]
            shares_per_calon[nomor][g].append((x,y))

    hasil = {}
    for nomor, data_per_g in shares_per_calon.items():
        berhasil = 0
        for g, shares in data_per_g.items():
            reconstructed = sss.reconstruct_data(shares, prime)
            expected = int(hashlib.sha256(f"{nama_calon_map[nomor]}_{g}".encode()).hexdigest(), 16)
            if reconstructed == expected:
                berhasil += 1
        hasil[nomor] = berhasil

    return hasil
```

Fig. 33. Fungsi untuk menghitung suara dengan Skema Hitung Kasar

Skema Hitung Kasar hanya menangani perhitungan sampai granularitas yang ditentukan. Skema gagal menentukan pemenang apabila selisih suara lebih kecil dari granularitas.

C. Implementasi Skema Hitung Halus

```python
def hitung_suara_halus(nama_calon_map, granularitas, n, prime, kotak="kotak_suara", pemecah="pemecah_buntu"):
    shares_aktif = kumpulkan_share(kotak, granularitas)

    hasil = hitung_berhasil(shares_aktif, nama_calon_map, granularitas, prime)
    # Hitung normal dulu
    print("Hasil awal: ")
    for nomor, berhasil in sorted(hasil.items()):
        print(f"{nama_calon_map[nomor]}: {berhasil}/{granularitas} pencacah terbuka")

    max_pencacah = max(hasil.values())
    pemimpin = [nomor for nomor, b in hasil.items() if b == max_pencacah]

    if len(pemimpin) == 1:
        return hasil, nama_calon_map[pemimpin[0]], False

    print(f"Seri antara: {[nama_calon_map[n] for n in pemimpin]}")
    print(f"Mulai binar search dengan pemecah_buntu")

    penghitung_terpakai = {nomor: 0 for nomor in pemimpin}
    pencacah_tertinggi = max_pencacah

    for g_target in range(pencacah_tertinggi+1, granularitas+1):
        print(f"Mengecek pencacah ke-{g_target}")

        while True:
            if all(penghitung_terpakai[nomor] > n for nomor in pemimpin):
                prime("Share penghitung habis, hasil seri")
                return hasil, None, True

            terbuka = []

            for nomor in pemimpin:
                if penghitung_terpakai[nomor] >= n:
                    continue

                nama = nama_calon_map[nomor]
                folder_pemecah = os.path.join(pemecah, f"shares_{nama}")
                penghitung_terpakai[nomor] += 1
                i = penghitung_terpakai[nomor]

                filepath = os.path.join(folder_pemecah, f"{nama}_{i}.json")
                with open(filepath, 'r') as f:
                    data = json.load(f)

                x = data[f"data_{g_target}"]["x"]
                y = data[f"data_{g_target}"]["y"]
                shares_aktif[nomor][g_target].append((x, y))

                reconstructed = sss.reconstruct_data(shares_aktif[nomor][g_target], prime)
                expected = int(hashlib.sha256(f"{nama}_{g_target}".encode()).hexdigest(), 16)

                if reconstructed == expected:
                    terbuka.append(nomor)

            if terbuka:
                print(f"+1 share penghitung tiap calon, total terpakai:")
                print(f"{ {nama_calon_map[n]: penghitung_terpakai[n] for n in pemimpin} }")
                print(f"Pencacah ke-{g_target} terbuka untuk:")
                print(f"{[nama_calon_map[n] for n in terbuka]}")

                if len(terbuka) == 1:
                    pemenang_nomor = terbuka[0]
                    hasil[pemenang_nomor] = g_target
                    return hasil, nama_calon_map[pemenang_nomor], False
                else:
                    for nomor in pemimpin:
                        hasil[nomor] = g_target
                    break

    print("Semua pencacah habis dan masih seri")
    return hasil, None, True
```

Fig. 34. Fungsi untuk menghitung suara dengan Skema Hitung Halus

Skema Hitung Halus mampu menangani perhitungan suara dengan presisi satuan.

V. PENGUJIAN

A. Pengujian Skema Naif

Kasus Berhasil (pemenang jelas):

```
==================
Selamat datang di Pemilu SSS
Sebuah pemilu kriptografis yang hanya memakai
Shamir Secret Sharing sebagai mekanik pemilu
Pilih skema yang ingin dipakai (1/2/3)
1. Skema Naif
2. Skema Hitung Kasar
3. Skema Hitung Halus
4. Keluar
pilihan: 1

================== Pemilu SSS ==================
Selamat datang di Pemilu SSS
Sebuah skema pemilihan umum yang hanya memakai Shamir Secret Sharing
Skema yang Anda jalankan adalah skema 1: Skema Naif
Pemilu hanya akan berhasil jika ada 1 calon
yang mendapat suara mayoritas sederhana (50%+1 dari total suara)

********************
Masukkan jumlah calon: 3
********************

********************
Masukkan jumlah pemilih: 100
********************

********************
Jumlah calon: 3
Jumlah pemilih: 100
********************

********************
Nama calon 1: Alice
Nama calon 2: Bob
Nama calon 3: Clara
********************

********************
Nama calon:
Calon ke-1: Alice
Calon ke-2: Bob
Calon ke-3: Clara
********************
```

Fig. 35. Inisialisasi parameter

```
********************
Data calon:
Calon ke-1: 280991720293
Calon ke-2: 4353890
Calon ke-3: 289581134433
********************

********************
Ambang batas k (50%+1): 51
Total share per calon: 100
********************

********************
100 share disimpan di folder: suara/shares_Alice
100 share disimpan di folder: suara/shares_Bob
100 share disimpan di folder: suara/shares_Clara
********************
```

Fig. 36. Persiapan surat suara

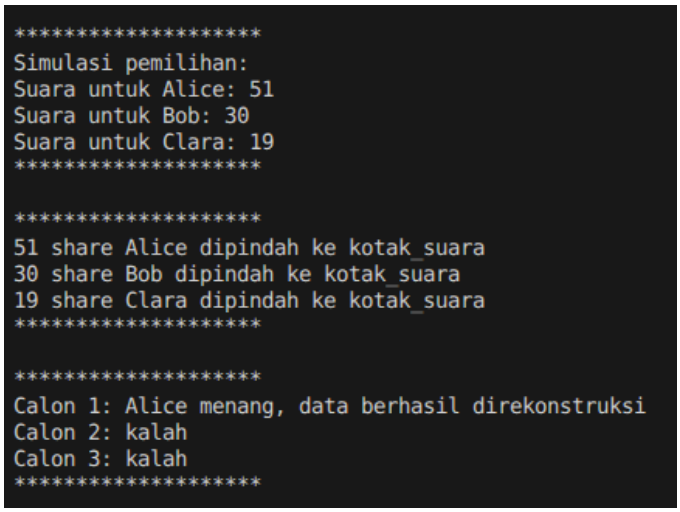


Fig. 37. Simulasi pemilihan

Kasus Gagal (gagal mayoritas):

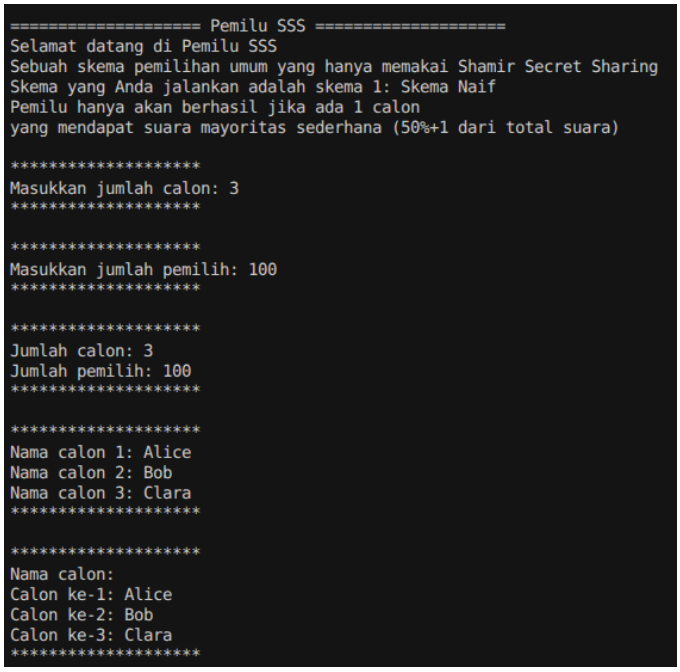


Fig. 38. Inisialisasi parameter

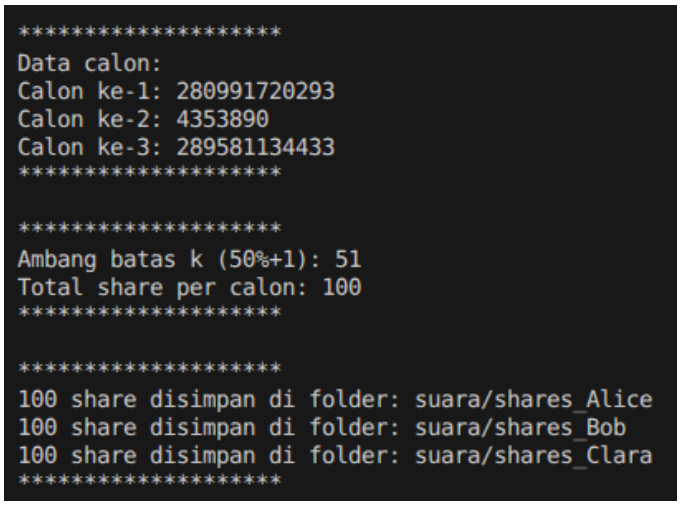


Fig. 39. Persiapan surat suara

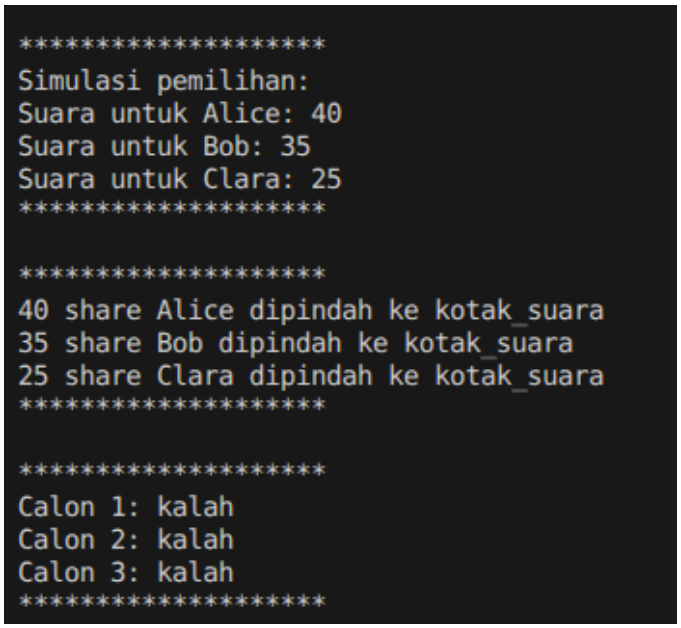


Fig. 40. Simulasi pemilihan

B. Pengujian Skema Hitung Kasar

Kasus Berhasil (selisih >= granularitas):

```
==================
Selamat datang di Pemilu SSS
Sebuah pemilu kriptografis yang hanya memakai
Shamir Secret Sharing sebagai mekanik pemilu
Pilih skema yang ingin dipakai (1/2/3)
1. Skema Naif
2. Skema Hitung Kasar
3. Skema Hitung Halus
4. Keluar
pilihan: 2

================== Pemilu SSS ==================
Selamat datang di Pemilu SSS
Sebuah skema pemilihan umum yang hanya memakai Shamir Secret Sharing
Skema yang Anda jalankan adalah skema 2: Skema Hitung Kasar
Hasil perolehan suara bisa dihitung sampai granularitas yang dipilih

********************
Masukkan jumlah calon: 3
********************

********************
Masukkan jumlah pemilih: 100
********************

********************
Masukkan granularitas (contoh: 10 = per 10%): 10
********************

********************
Nama calon 1: Alice
Nama calon 2: Bob
Nama calon 3: Clara
********************
```

Fig. 41. Inisialisasi parameter

```
**********************
Batas per pencacah (data granular):
  data_1: k=10 (10%)
  data_2: k=20 (20%)
  data_3: k=30 (30%)
  data_4: k=40 (40%)
  data_5: k=50 (50%)
  data_6: k=60 (60%)
  data_7: k=70 (70%)
  data_8: k=80 (80%)
  data_9: k=90 (90%)
  data_10: k=100 (100%)
**********************

**********************
100 share disimpan di folder: suara/shares_Alice
100 share disimpan di folder: suara/shares_Bob
100 share disimpan di folder: suara/shares_Clara
**********************
```

Fig. 42. Persiapan surat suara

```
********************
Simulasi pemilihan:
Suara untuk Alice: 50
Suara untuk Bob: 40
Suara untuk Clara: 10
********************

********************
50 share Alice dipindah ke kotak_suara
40 share Bob dipindah ke kotak_suara
10 share Clara dipindah ke kotak_suara
********************

********************
Calon Alice: 5/10 data berhasil dibuka, (berhasil dapat setidaknya 50% suara)
Calon Bob: 4/10 data berhasil dibuka, (berhasil dapat setidaknya 40% suara)
Calon Clara: 1/10 data berhasil dibuka, (berhasil dapat setidaknya 10% suara)
********************
```

Fig. 43. Simulasi pemilihan

Kasus Gagal (selisih <= granularitas):

```
================== Pemilu SSS ==================
Selamat datang di Pemilu SSS
Sebuah skema pemilihan umum yang hanya memakai Shamir Secret Sharing
Skema yang Anda jalankan adalah skema 2: Skema Hitung Kasar
Hasil perolehan suara bisa dihitung sampai granularitas yang dipilih

********************
Masukkan jumlah calon: 3
********************

********************
Masukkan jumlah pemilih: 100
********************

********************
Masukkan granularitas (contoh: 10 = per 10%): 10
********************

********************
Nama calon 1: Alice
Nama calon 2: Bob
Nama calon 3: Clara
********************
```

Fig. 44. Inisialisasi parameter

```
**********************
Batas per pencacah (data granular):
  data_1: k=10 (10%)
  data_2: k=20 (20%)
  data_3: k=30 (30%)
  data_4: k=40 (40%)
  data_5: k=50 (50%)
  data_6: k=60 (60%)
  data_7: k=70 (70%)
  data_8: k=80 (80%)
  data_9: k=90 (90%)
  data_10: k=100 (100%)
**********************

**********************
100 share disimpan di folder: suara/shares_Alice
100 share disimpan di folder: suara/shares_Bob
100 share disimpan di folder: suara/shares_Clara
**********************
```

Fig. 45. Persiapan surat suara

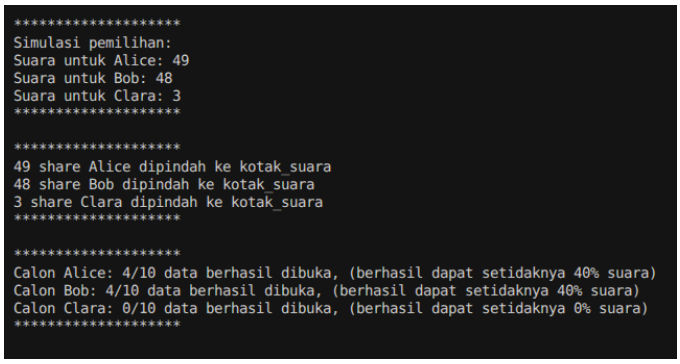


Fig. 46. Simulasi pemilihan

C. Pengujian Skema Hitung Halus

Kasus Pemenang Mayoritas:

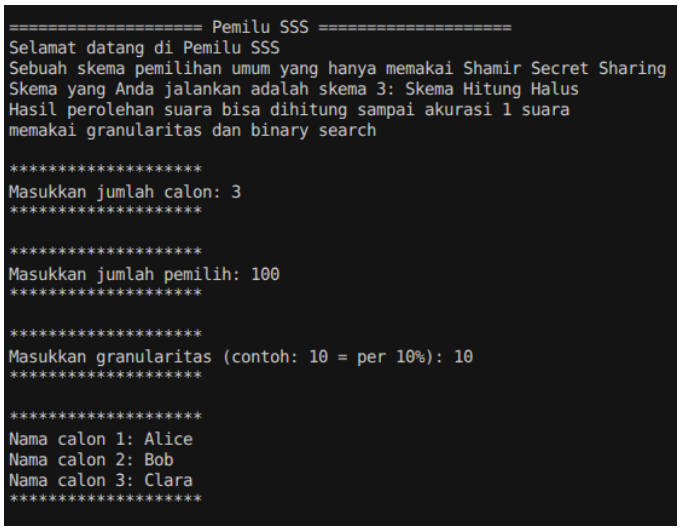


Fig. 47. Inisialisasi parameter

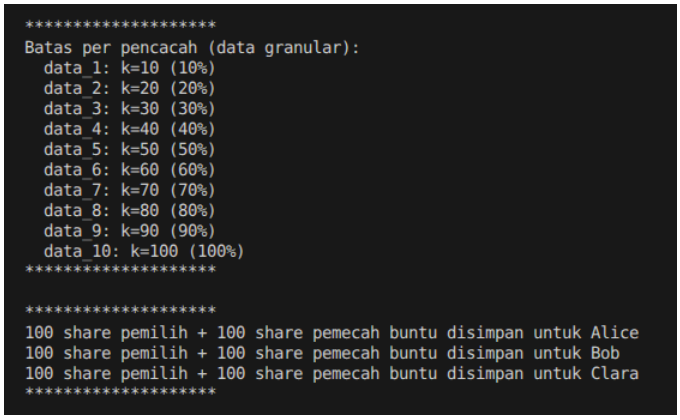


Fig. 48. Persiapan surat suara

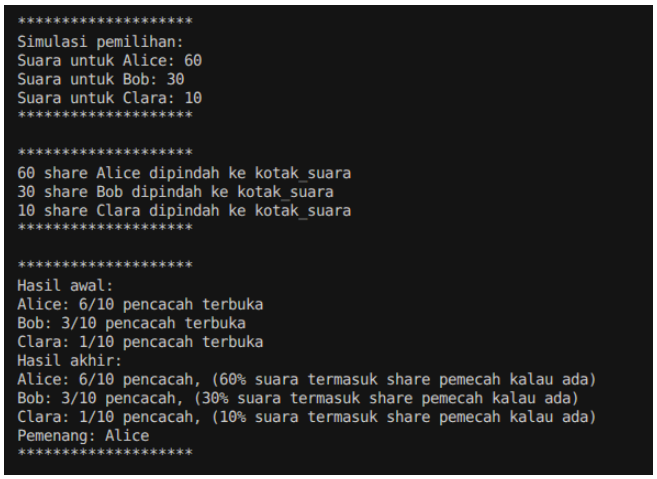


Fig. 49. Simulasi pemilihan

Kasus Pemenang Granular:

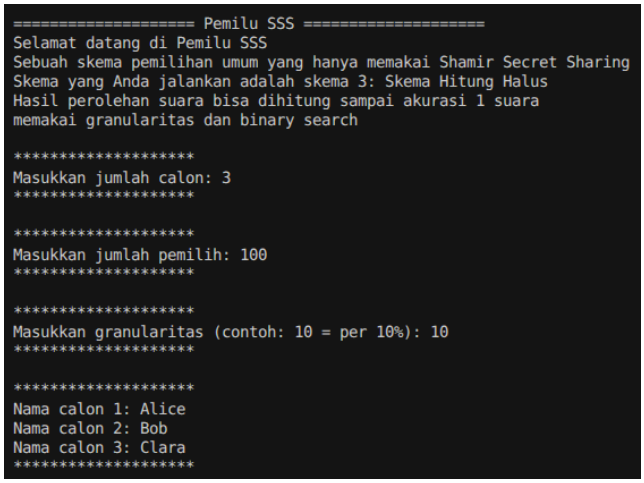


Fig. 50. Inisialisasi parameter

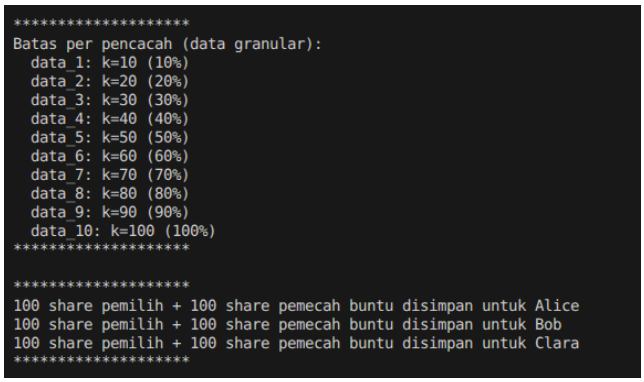


Fig. 51. Persiapan surat suara

```
********************
Simulasi pemilihan:
Suara untuk Alice: 50
Suara untuk Bob: 40
Suara untuk Clara: 10
********************

********************
50 share Alice dipindah ke kotak_suara
40 share Bob dipindah ke kotak_suara
10 share Clara dipindah ke kotak_suara
********************

********************
Hasil awal:
Alice: 5/10 pencacah terbuka
Bob: 4/10 pencacah terbuka
Clara: 1/10 pencacah terbuka
Hasil akhir:
Alice: 5/10 pencacah, (50% suara termasuk share pemecah kalau ada)
Bob: 4/10 pencacah, (40% suara termasuk share pemecah kalau ada)
Clara: 1/10 pencacah, (10% suara termasuk share pemecah kalau ada)
Pemenang: Alice
********************
```

Fig. 52. Simulasi pemilihan

Kasus Pemenang Upa-granular:

```
================== Pemilu SSS ==================
Selamat datang di Pemilu SSS
Sebuah skema pemilihan umum yang hanya memakai Shamir Secret Sharing
Skema yang Anda jalankan adalah skema 3: Skema Hitung Halus
Hasil perolehan suara bisa dihitung sampai akurasi 1 suara
memakai granularitas dan binary search

********************
Masukkan jumlah calon: 3
********************

********************
Masukkan jumlah pemilih: 100
********************

********************
Masukkan granularitas (contoh: 10 = per 10%): 10
********************

********************
Nama calon 1: Alice
Nama calon 2: Bob
Nama calon 3: Clara
********************
```

Fig. 53. Inisialisasi parameter

```
********************
Batas per pencacah (data granular):
  data_1: k=10 (10%)
  data_2: k=20 (20%)
  data_3: k=30 (30%)
  data_4: k=40 (40%)
  data_5: k=50 (50%)
  data_6: k=60 (60%)
  data_7: k=70 (70%)
  data_8: k=80 (80%)
  data_9: k=90 (90%)
  data_10: k=100 (100%)
********************

********************
100 share pemilih + 100 share pemecah buntu disimpan untuk Alice
100 share pemilih + 100 share pemecah buntu disimpan untuk Bob
100 share pemilih + 100 share pemecah buntu disimpan untuk Clara
********************
```

Fig. 54. Persiapan surat suara

```
********************
Simulasi pemilihan:
Suara untuk Alice: 49
Suara untuk Bob: 48
Suara untuk Clara: 3
********************

********************
49 share Alice dipindah ke kotak_suara
48 share Bob dipindah ke kotak_suara
3 share Clara dipindah ke kotak_suara
********************

********************
Hasil awal:
Alice: 4/10 pencacah terbuka
Bob: 4/10 pencacah terbuka
Clara: 0/10 pencacah terbuka
Seri antara: ['Alice', 'Bob']
Mulai perhitungan dengan pemecah buntu
Mengecek pencacah ke-5
+1 share penghitung tiap calon, total terpakai:
{'Alice': 1, 'Bob': 1}
Pencacah ke-5 terbuka untuk:
['Alice']
Hasil akhir:
Alice: 5/10 pencacah, (50% suara termasuk share pemecah kalau ada)
Bob: 4/10 pencacah, (40% suara termasuk share pemecah kalau ada)
Clara: 0/10 pencacah, (0% suara termasuk share pemecah kalau ada)
Pemenang: Alice
********************
```

Fig. 55. Simulasi pemilihan

Kasus Seri:

```
================== Pemilu SSS ==================
Selamat datang di Pemilu SSS
Sebuah skema pemilihan umum yang hanya memakai Shamir Secret Sharing
Skema yang Anda jalankan adalah skema 3: Skema Hitung Halus
Hasil perolehan suara bisa dihitung sampai akurasi 1 suara
memakai granularitas dan binary search

********************
Masukkan jumlah calon: 3
********************

********************
Masukkan jumlah pemilih: 100
********************

********************
Masukkan granularitas (contoh: 10 = per 10%): 10
********************

********************
Nama calon 1: Alice
Nama calon 2: Bob
Nama calon 3: Clara
********************
```

Fig. 56. Inisialisasi parameter

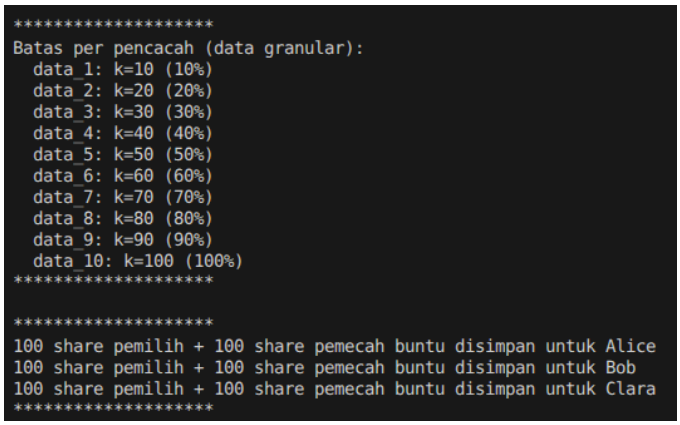


Fig. 57. Persiapan surat suara

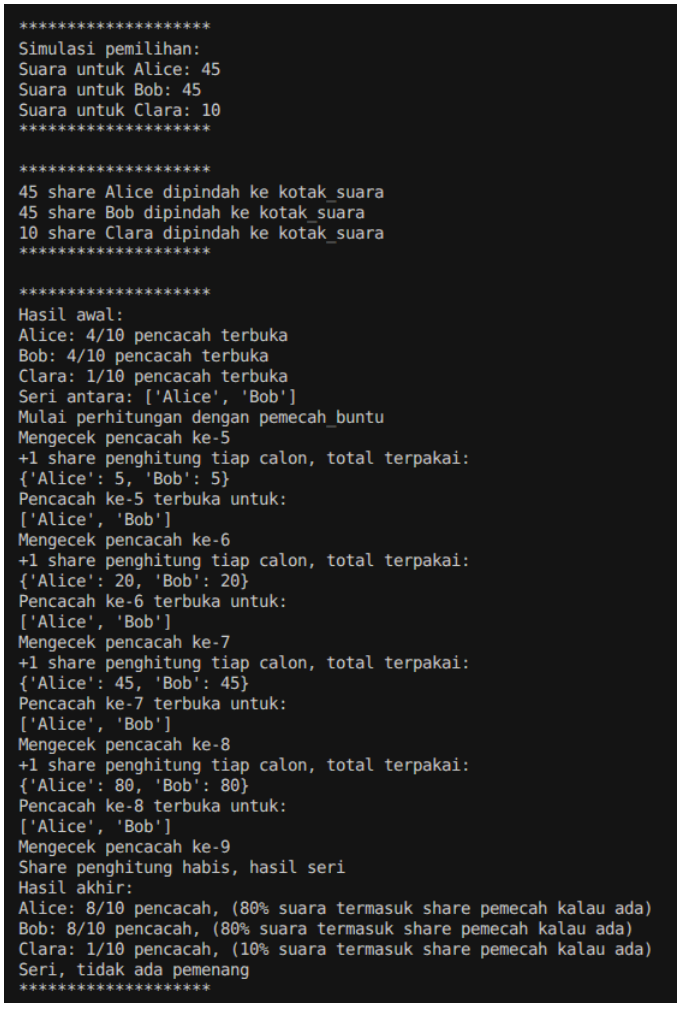


Fig. 58. Simulasi pemilihan

VI. Analisis Kelayakan

A. *Analisis Manfaat*

Pemilu SSS menegakkan beberapa prinsip luber-jurdil berikut:

1. Rahasia:

   Semua surat suara hanya terlihat sebagai rangkaian angka acak. Panitia tidak akan bisa mengetahui siapa yang memilih siapa. Kerahasiaan pilihan terjaga.

2. Jujur:

   Kecurangan dari pihak panitia bisa diminimalisasi. Tujuan surat suara (calon yang dipilih dengan suatu surat suara) tidak bisa diketahui sampai perhitungan dilakukan. Manipulasi bisa berpotensi merusak suara untuk calon yang ingin dimenangkan oleh pelaku kecurangan. Hal ini bertindak sebagai ancaman pencegah (*deterrent*).

3. Adil:

   Apabila kecurangan lebih sulit dilakukan, pemilu akan lebih adil.

B. *Analisis Kompleksitas Waktu*

Kompleksitas waktu yang diperlukan untuk perhitungan suara utamanya dipengaruhi kompleksitas Interpolasi Lagrange. Proses ini memiliki kompleksitas $O(n^2)$. Untuk penerapan di Indonesia, perhitungannya adalah sebagai berikut:

1. Ada 200 juta pemilih (anggaplah pemilih di Indonesia berada di angka tersebut)
2. NVIDIA 4090 mampu untuk melakukan sekitar 8×10^{13} flops per detik
3. NVIDIA 4090 memiliki harga \$1600 (kira-kira 30 juta rupiah)
4. $O(n^2) = (2 \times 10^8)^2 = 4 \times 10^{16}$ flops
5. Waktu yang dibutuhkan untuk menghitung hasil dengan 1 buah NVIDIA 4090: $4 \times 10^{16} \div 8 \times 10^{13} = 0.5 \times 10^3 = 500$ detik

Kompleksitas waktu untuk memakai skema ini masih bisa ditoleransi. Apalagi mengingat biaya pemilu biasanya berada di angka triliunan.

VII. Kekurangan & Pengembangan Lanjutan

A. *Kekurangan*

Kekurangan utama dari skema ini adalah skema mengasumsikan server yang aman. Shamir Secret Sharing memerlukan dealer yang terpercaya agar bisa menjaga rahasia. Dalam skema ini, dealer adalah komputer server yang menjalankan programnya. Apabila terdapat pelaku kecurangan di dalam panitia yang memiliki kemampuan teknis cukup, skema ini bisa dibobol dan pemilu bisa dicurangi.

B. *Kesempatan Pengembangan Lanjutan*

Diperlukan lapisan pertahanan lain untuk mengamankan pemilu. Mengamankan proses perhitungan suara saja tidak cukup, proses pembagian dan pemungutan suara juga harus diamankan. Hal lain yang bisa ditambahkan selain lapisan lain adalah implementasi binary search di Skema Hitung Halus. Saat ini penambahan share penghitung masih dilakukan secara linear sehingga kompleksitasnya linear juga. Semua hal ini diharapkan bisa dikembangkan lebih lanjut di masa depan.

Tautan Rilis di Github

https://github.com/RafiHalliday/pemilu-shamir-secret-sharing/releases/tag/Pemilu_SSS

Ucapan Terima Kasih

Saya mengucapkan terima kasih kepada orangtua, teman, dan pak Rinaldi Munir yang telah membantu saya dalam pengerjaan makalah ini. Berkat bantuan ilmu dan dukungan dari mereka, makalah ini berhasil diselesaikan.

Referensi

[1] NVIDIA, "NVIDIA GeForce RTX 4090 graphics cards," NVIDIA. https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4090/

[2] TechPowerUp, "NVIDIA GeForce RTX 4090 Specs," TechPowerUp, Oct. 31, 2022. https://www.techpowerup.com/gpu-specs/geforce-rtx-4090.c3889

[3] GeeksforGeeks, "Implementing Shamir's Secret Sharing Scheme in Python," GeeksforGeeks, Apr. 27, 2020. https://www.geeksforgeeks.org/python/implementing-shamirs-secret-sharing-scheme-in-python/

[4] C. Jeffrey, "ThumbsUp -A Secure Voting System: Design and Implementation with Shamir's Secret Sharing Leveraging Homomorphic Property of Lagrange Interpolation."

[5] C. Sadewa, "A Freely Modifiable Secret Sharing An Extension to Shamir Secret Sharing for Achieving Independtly Modifiable Secret Share."

[6] Badan Pusat Statistik Indonesia, "Jumlah Pemilih - Tabel Statistik," Bps.go.id, 2023. https://www.bps.go.id/id/statistics-table/2/MTY1IzI=/jumlah-pemilih.html

[7] "UU No. 3 Tahun 1999," Database Peraturan | JDIH BPK, 2026. https://peraturan.bpk.go.id/Details/45271 (accessed Jun. 19, 2026).

[8] emcapsulation, "Protecting Your Secrets With Polynomials - Shamir's Secret Sharing," YouTube, Aug. 18, 2025. https://www.youtube.com/watch?v=_EHMd8gpApo (accessed Jun. 07, 2026).

[9] Andika Dwi and L. Trianita, "6 Dugaan Kecurangan Pemilu 2024, dari Server Sirekap Hingga Surat Suara Telah Tercoblos," Tempo, Feb. 19, 2024. https://www.tempo.co/hukum/6-dugaan-kecurangan-pemilu-2024-dari-server-sirekap-hingga-surat-suara-telah-tercoblos-85859 (accessed Jun. 19, 2026).

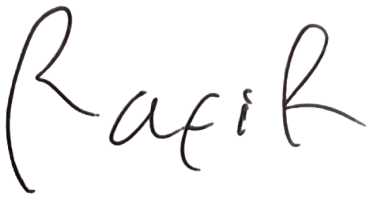
[10] Laudia Tysara, "10 Kasus Pemilu di Indonesia 2004-2024, Pelanggaran Terjadi Berulang," liputan6.com, Feb. 15, 2024. https://www.liputan6.com/hot/read/5528538/10-kasus-pemilu-di-indonesia-2004-2024-pelanggaran-terjadi-berulang

[11] A. Rais, "Analisis Protokol Suara Mayoritas Terdesentralisasi Berbasis Pembagian Rahasia Shamir." Accessed: Jun. 19, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Makalah-UAS/Makalah-UAS-Kripto-2020%20(35).pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026



Muhammad Rafi Ramadhan