Implementasi dan Analisis Protokol *Verifiable Secret Sharing Scheme*

Farras Mohammad Hibban Faddila - 13518017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518017@std.stei.itb.ac.id

***Abstrak*—*Secret Sharing Scheme* merupakan skema pembagian rahasia pada bidang kriptografi yang digunakan bagi seseorang untuk membagi rahasia kepada beberapa pihak, tanpa pihak tersebut dapat mengetahui rahasia tersebut secara individual. Skema ini merupakan bagian utama dari *threshold cryptography*, dan digunakan atas dasar *confidentiality serta integrity data.* Terdapat ekstensi dari *secret sharing scheme*, yaitu verifiable secret sharing scheme (VCSS), yang mengatasi permasalahan terdapat pihak-pihak yang tidak jujur. Pada makalah ini akan dijelaskan implementasi dari dua buah algoritma pada VCSS, yakni algoritma feldman dan algoritma Pedersen, serta dilakukan perbandingan antara kedua algoritma tersebut.**

***Kata kunci*—Verifiable, Skema Pembagian Rahasia, Pedersen, Feldman, Shamir**

I. PENDAHULUAN

Secara istilah, kriptografi merupakan ilmu dan seni untuk menjaga keamanan pesan. Sedangkan secara bahasa, kriptografi berasal dari dua kata dalam bahasa Yunani, yakni cryptos yang berarti rahasia, serta graphein yang berarti tulisan, sehingga secara literal kriptografi dapat diartikan sebagai tulisan rahasia. Teknik umum pada kriptografi adalah melakukan enkripsi (penyandian) pada pesan yang akan dikirim, dan ketika pesan sampai kepada penerima, dilakukan dekripsi, yaitu pengembalian pesan ke dalam bentuk semula. Untuk melakukan kedua proses ini, dibutuhkan kunci rahasia yang disepakati oleh pengirim dan penerima terlebih dahulu, sehingga pihak ketiga yang menyadap pesan saat pesan sedang berada pada saluran pengiriman tidak dapat mengerti pesan tersebut, kecuali memiliki kunci yang sama.

Seiring waktu, kriptografi berkembang dan berperan penting dalam sejarah dunia, contohnya pada perang dunia ke-2. Seiring berkembangnya teknik-teknik pada kriptografi, berkembang juga ilmu kriptanalisis, yakni ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui kunci yang digunakan. Selain itu, muncul juga skema enkripsi asimetrik yang mengandalkan kunci publik dan kunci privat.

Pada kriptografi kunci publik, *sender* akan men-*generate* sebuah pasangan kunci, yakni kunci privat dan kunci publiknya. Kunci privat digunakan untuk melakukan enkripsi pada pesan, dan kunci publik digunakan untuk dekripsi pesan yang terenkripsi. Kunci publik dapat dibagikan, tetapi kunci privat harus dijaga kerahasiaannya.

Pada beberapa situasi, terkadang lebih diinginkan agar kunci privat tidak hanya dijaga kerahasiannya oleh satu orang, melainkan dijaga oleh lebih dari satu orang. Sebagai contoh, sebuah brankas pada bank dibuka oleh dua buah kunci yang masing-masing dijaga oleh pemilik brankas dan pegawai bank. Situasi-situasi seperti ini menginginkan agar kepemilikan (*ownership*) atau pengetahuan (*knowledge*) dari kunci privat tersebut didistribusikan kepada sejumlah pihak tambahan. Salah satu alasannya adalah untuk menghindari peluang terjadinya *loss of key*. Selain itu, dengan mendistribusikan kunci kepada sejumlah orang, maka akan lebih sulit untuk melakukan perubahan pada nilai kunci tersebut, dengan asumsi bahwa seluruh pihak pemegang kunci tidak saling bekerja sama. Dengan kata lain, kunci yang disimpan menjadi lebih *reliable*.

Untuk mengimplementasikan hal tersebut, terdapat bagian dalam kriptografi yang disebut sebagai *Threshold Cryptography*. Pada bagian ini dipelajari teknik-teknik untuk melakukan distribusi sebuah rahasia pada beberapa pihak, dengan catatan sejumlah pihak (sebanyak *threshold*) mampu merekonstruksi rahasia tersebut. Teknik pendistribusian pesan tersebut disebut dengan Skema Pembagian Rahasia. Dalam skema pembagian rahasia terdapat seorang *dealer* (distributor rahasia), serta beberapa orang *participant* (pihak pemegang rahasia). Sejumlah *participant* manapun yang berjumlah kurang dari *threshold* tidak dapat merekonstruksi rahasia dari bagian-bagian yang diberikan kepada mereka, sedangkan sejumlah *participant* apapun yang berjumlah tidak kurang dari *threshold* dapat merekonstruksi rahasia tersebut.

Dalam perkembangannya, muncul beberapa parameter tambahan dalam skema pembagian rahasia seperti kejujuran *dealer* maupun kejujuran *participant. Participant* bisa saja sewaktu-waktu mengubah nilai yang diberikan kepadanya sehingga rahasia tidak dapat direkonstruksi. Selain itu, *dealer* juga bisa saja dapat memberikan nilai rahasia yang telah diubah, sehingga *participant* tidak dapat merekonstruksi rahasia, dan muncul kecurigaan kepada *participant* kalau-kalau *participant* mengubah nilai dari rahasia yang diberikan. Sebuah skema tambahan dibutuhkan untuk menghindari hal ini, dan dalam makalah ini akan dibahas mengenai dua algoritma *secret sharing* yang mengatasi hal tersebut, yaitu algoritma Feldman dan algoritma Pedersen. Kedua algoritma tersebut disebut juga sebagai *Verifiable Secret Sharing*, yaitu skema *secret sharing* di mana tiap *holder* dapat melakukan verifikasi terhadap *dealer* bahwa nilai yang diberikan sudah benar.

II. DASAR TEORI

A. Skema Pembagian Rahasia

Skema Pembagian Rahasia merupakan sebuah metode untuk mendistribusikan sebuah rahasia (dinotasikan dengan s) kepada beberapa pihak yang disebut sebagai *participant*, dan setiap *participant* memperoleh bagian dari rahasia yang diberikan, yaitu *share*. Rahasia yang disembunyikan tersebut hanya dapat direkonstruksi dari *share-share* milik para *participant* apabila jumlah *share* yang digunakan untuk melakukan rekonstruksi melebihi sebuah nilai ambang (*threshold*), yang selanjutnya akan dinotasikan sebagai t. Jumlah *participant* yang menerima *share* dinotasikan sebagai n. *Participant* ke-i menerima sebuah *share* yang dinotasikan sebagai s_i. Kedua nilai t dan n tersebut merupakan parameter dari sebuah skema pembagian rahasia, dan sistem ini disebut juga sebagai *(t,n)-threshold scheme.*

Berikut ini merupakan beberapa skema pembagian rahasia untuk nilai t yang spesial. Jika t = 1, maka setiap *participant* dapat merekonstruksi rahasia yang diberikan tanpa perlu menggabungkan *share* miliknya dengan *share* milik *participant* lain. Hal ini berarti cukup distribusikan saja rahasia tersebut kepada seluruh *participant* apa adanya (s = s_1 = s_2 = … = s_n). Untuk t = n, salah satu caranya adalah memberikan n – 1 *participant* sebuah bilangan acak p_i untuk 1 <= i <= n – 1 sebagai *share*-nya, dan memberikan *participant* terakhir nilai dari s xor p_1 xor p_2 … xor p_(n-1) sebagai *share*-nya.

Beberapa penggunaan Skema Pembagian Rahasia dalam dunia industri adalah penyimpanan nomor akun bank, penyimpanan kunci enkripsi (kunci simetri ataupun kunci privat), serta distribusi kunci pada beberapa server di sebuah *environment cloud computing*.

Skema Pembagian Rahasia menjawab dua buah aspek pada kriptografi yakni *confidentiality* serta *reliability*. Data tentunya tidak boleh bocor kepada pihak lain, namun yang tidak kalah penting, data tidak boleh hilang. Metode tradisional tidak dapat mencapai *high level of confidentiality* serta *high level of reliability* sekaligus. Hal ini karena, untuk mencapai *high level of confidentiality*, rahasia harus disimpan pada sesedikit mungkin lokasi, namun hal ini meningkatkan potensi data hilang (karena tidak ada *backup*), sehingga *reliability* rendah. Sedangkan, untuk mencapai *high level of reliability*, rahasia perlu disimpan pada beberapa tempat, tetapi hal ini meningkatkan peluang data bocor.

B. Skema Pembagian Rahasia Shamir

Skema Pembagian Rahasia Shamir merupakan sebuah *Secret Sharing Scheme* yang dikembangkan oleh Adi Shamir. Dalam skema ini, t dapat bernilai apa saja pada range 1 hingga n. Sistem ini memanfaatkan fakta bahwa pada t buah titik yang berbeda pada bidang, terdapat secara unik sebuah polinomial berderajat t – 1 yang melalui seluruh titik tersebut.

Skema Pembagian Rahasia Shamir ((t,n)-threshold) didefinisikan sebagai berikut ini

Terdapat beberapa karakteristik dari Skema Pembagian Rahasia Shamir sebagai berikut:

a) Secure
   Skema ini aman berdasarkan teori informasi
b) Minimal
   Ukuran dari tiap *share* tidak melebihi ukuran dari data awal
c) Extensible
   Apabila nilai threshold tetap, partisipan dapat ditambah atau dikurang tanpa memengaruhi *share* yang masih ada
d) Dynamic
   Tingkat keamanan dapat ditingkatkan dengan mengubah polinomial dan memberikan *share* baru kepada *participant.*
e) Flexible
   Tiap *participant* dapat menerima sejumlah *share* yang berbeda-beda, dan hal ini dapat dimanfaatkan untuk mengimplementasikan tingkat kepentingan dari *participant*, contohnya seorang *participant* A dapat merekonstruksi rahasia secara sendiri karena diberikan *share* sejumlah *threshold*, sedangkan *participant* B tidak bisa karena diberikan *share* sejumlah kurang dari *threshold.*

B. Skema Pembagian Rahasia Terverifikasi

Skema Pembagian Rahasia Terverifikasi pada umumnya merupakan sebuah skema pembagian rahasia yang memiliki sebuah protocol tambahan bagi *participant* untuk melakukan verifikasi atas *share* yang diterima. Skema ini dirancang untuk dapat menghindari dua jenis *attack* berikut ini:

a) *Dealer* sengaja mengirim *share* yang tidak tepat kepada semua atau beberapa *participant* selama pendistribusian
b) *Participant* menggunakan *share* yang tidak tepat saat proses rekonstruksi.

Umumnya terdapat dua jenis Verifiable Secret Sharing Schema yang akan dibahas pada makalah ini.

1) Skema Feldman
   Skema ini menggunakan skema pembagian rahasia shamir dalam melakukan pendistribusian *share*, menggunakan polinomial berikut ini.

$$a(X) = s + u_1 X + \cdots + u_t X^t,$$

   dan bilangan prima p sebagai modulus (operasi dilakukan pada lapangan Z_p)

   Berikutnya, untuk melakukan verifikasi, digunakan sebuah bilangan basis g yang merupakan generator pada lapangan prima q (di share secara public ke partisipan), di mana q merupakan sebuah prima sehingga p membagi q-1. Bilangan q ini dipilih atas dasar sulitnya mencari logaritma diskrit. Setiap participant diberikan nilai $B_j = g^{u_j}$, di mana u_j merupakan koefisien dari polinomial yang digunakan untuk membangkitkan share. Misalkan participant ke-i menggunakan titik x_i dalam penghitungan sharenya. Setelah itu, untuk melakukan verifikasi, maka participant ke-i tersebut cukup menghitung nilai dari:

$$\prod_{j=0}^{t} B_j^{ij}.$$

Bentuk tersebut ekuivalen dengan perkalian dari $g^{u_j i^j}$ untuk setiap j, dan jika dikalikan untuk setiap j, maka pangkatnya akan dijumlahkan menjadi a(i), yang merupakan nilai share milik participant ke-i. Sehingga, nilai tersebut cukup dibandingkan apakah sama dengan nilai dari g^{s_i}, di mana s_i merupakan share yang diterima oleh participant ke i. Apabila hasilnya sama, maka share tersebut merupakan share yang valid dan berhasil diverifikasi.

2) Skema Pedersen

Perbedaan utama skema Pedersen dengan skema feldman terletak pada jumlah polinomial yang digunakan. Skema Pedersen menggunakan dua buah polinomial, sehingga setiap participant menerima dua buah share dalam bentuk dua pasang bilangan. Polinomial pertama memiliki suku konstanta berupa s (nilai rahasia), sedangkan polinomial kedua tidak perlu.

$$a(X) = u_0 + u_1 X + \cdots + u_t X^t$$
$$b(X) = v_0 + v_1 X + \cdots + v_t X^t,$$

Setiap partisipan ke-i menerima share berupa pasangan (a(i), b(i)). Selain itu, *dealer* juga mengumuman kepada seluruh partisipan nilai dari komitmen C_0, C_1, … C_t yang dikalkulasi sebagai berikut:

$$C_j = g^{u_j} h^{v_j},\ 0 \le j \le t.$$

Untuk melakukan verifikasi dari share yang dibagikan, partisipan ke- cukup melakukan kalkulasi dari ekspresi berikut ini. $((s_{i1}, s_{i2})$ merupakan share yang dimiliki participant ke i)

$$g^{s_{i1}} h^{s_{i2}} = \prod_{j=0}^{t} C_j^{ij}.$$

Ketika bentuk tersebut diekspansi, nilainya akan menjadi

$$g^{u_0 + u_1 i + u_2 i^2 + \cdots + u_t i^t} h^{v_0 + v_1 i + v_2 i^2 + \cdots + v_t i^t} = g^{a(i)} h^{b(i)}$$

yang akan valid apabila kedua ruas bernilai sama, karena a(i), b(i) merupakan share yang seharusnya diberikan oleh *dealer* kepada *participant*, dan jika kedua ruas bernilai sama maka akibatnya $(s_{i1}, s_{i2}) = (a(i), b(i))$

III. Implementasi

A. Rancangan Umum

Untuk mengimplementasi protocol *verifiable secret sharing scheme* digunakan *Shamir Secret Sharing Scheme* yang memiliki sebuah parameter yang menunjukkan apakah skema tersebut *verifiable*, dan jika skema tersebut *verifiable* maka apakah menggunakan skema Feldman atau skema Pedersen. Terdapat dua buah kelas, yakni kelas ShamirDealer serta kelas ShamirHolder yang memiliki struktur sebagai berikut.

```
class ShamirDealer:
    def __init__(self, t, n, s=None, p=None,
vss="Feldman"):
        """
        Inisialisasi sebuah dealer dengan
parameter t = threshold, n = jumlah partisipan,
s = secret, p = prima sebagai lapangan Zp yang
digunakan, dan vss sebagai parameter algoritma
verifiable secret sharing
        """
        self.secret = s
        self.numholders = n
        self.threshold = t
        if p == None:
            cur = sgp[-1]
            if cur < s or cur < n:
                raise Exception("Number of
holder or the secret is too big")
            self.prime = cur
            pass
        else:
            self.prime = p
        self.vss = vss
        if vss == "Feldman":
            self.polynomial =
self.generate_polynomial(c="secret")
        elif vss == "Pedersen":
            self.polynomial =
(self.generate_polynomial(c="secret"),
self.generate_polynomial())
        self.holders = self.generate_x()

    def set_secret(self, s):
        if s > self.prime:
            raise "Secret is too long to be
shared"
        self.secret = s

    def get_prime(self):
        return self.prime

    def generate_x(self):
        """
        Melakukan pembangkitan koordinat x dari
titik-titik kalkulasi share untuk tiap
partisipan
        """
        x_coordinates = []
        for i in range(self.numholders):
            #append random generated xi for
each participants
            #all different
            a = random.randint(0, self.prime -
1)
            while a in x_coordinates:
                a = random.randint(0,
```

```
self.prime - 1)
            x_coordinates.append(a)
        return x_coordinates

    def generate_polynomial(self, c=None):
        """
        Melakukan pembangkitan polinomial yang
digunakan pada pembuatan share. Apabila vss =
'Pedersen', maka dibangkitkan dua buah
polinomial
        """
        if c == "secret":
            koef = [self.secret]
        else:
            koef = [random.randint(0,
self.prime - 1)]
        for i in range(1, self.threshold):
            #generate random integers between 0
and p - 1 inclusive
            a = random.randint(0, self.prime -
1)
            koef.append(a)
        return Polynomial(koef, self.prime)

    def generate_share(self):
        """
        Membuat sebanyak n buah share dari
titik koordinat x serta polinomial yang sudah
dibangkitkan. Apabila vss = 'Pedersen', maka
dibangkitkan dua buah share
        """
        shares = []
        if self.vss == "Feldman":
            for point in self.holders:
                share =
self.polynomial.calc(point)
                shares.append((point, share,
self.prime))
        elif self.vss == "Pedersen":
            for point in self.holders:
                share0 =
self.polynomial[0].calc(point)
                share1 =
self.polynomial[1].calc(point)
                shares.append((point, (share0,
share1), self.prime))
        self.shares = shares

    def generate_commitments(self):
        """
        Membuat sebanyak n buah komitmen yang
dibagikan kepada partisipan untuk validasi
nilai dari share yang diberikan kepada mereka
        """
        p = self.prime
        q = 2 * p + 1
        g = semi_primitive_root(q)
        if self.vss == "Feldman":
            koefs =
self.polynomial.coefficients
            commits = [pow(g, koef, q) for koef
in koefs]
            h = None
            return Commit(commits, g, h, q,
self.vss)
        elif self.vss == "Pedersen":
            koefs0 =
self.polynomial[0].coefficients
            koefs1 =
self.polynomial[1].coefficients
            h = semi_primitive_root(q)
            while h == g:
                h = semi_primitive_root(q)
            commits = [(pow(g, koefs0[i], q) *
pow(h, koefs1[i], q) % q) for i in
range(len(koefs0))]
            return Commit(commits, g, h, q,
self.vss)

    def distribute(self):
        """
        Melakukan distribusi dari share dan
komitmen yang telah dibuat
        """
        self.generate_share()
        return [
            ShamirHolder(self.shares[i][0],
self.shares[i][1], self.prime) for i in
range(len(self.shares))
        ]
```

```
class ShamirHolder:
    def __init__(self, point, share, prime,
vss=None):
        #share is the y coordinate, point is the
x coordinate
        #prime is the prime used for the field
        self.point = point
        self.share = share
        self.prime = prime
        self.vss = vss
```

Selain itu, dibuat pula sebuah struktur data untuk mengimplementasikan komitmen untuk validasi sebagai berikut. Parameter h dibutuhkan pada skema Pedersen.

```
class Commit:
    def __init__(self, commits, g, h, p,
vss="Feldman"):
        self.generator = g
        self.prime = p
        self.commits = commits
        self.vss = vss
        if vss == "Pedersen":
            self.other = h
```

Terakhir, diimplementasi dua buah fungsi utama pada skema pembagian rahasia terverifikasi, yakni rekonstruksi rahasia serta verifikasi *share* sebagai berikut. Secara umum fungsi validasi akan melakukan perhitungan sesuai dengan yang telah diberikan pada bagian Skema Pembagian Rahasia Terverifikasi untuk setiap algoritma, sedangkan fungsi rekonstruksi memanfaatkan

interpolasi lagrange yang telah disebutkan juga pada pembahasan mengenai Skema Pembagian Rahasia Shamir.

```
def validate(commit, holder):
    point = holder.point
    share = holder.share
    commits = commit.commits
    vss = commit.vss
    p = commit.prime
    g = commit.generator
    if vss == "Feldman":
        cek1 = 1
        for i in range(len(commits)):
                cek1 = (cek1 * pow(commits[i],
pow(point, i, p - 1), p)) % p
        cek2 = pow(g, share, p)
        return cek1 == cek2
    elif vss == "Pedersen":
        cek1 = 1
        for i in range(len(commits)):
                cek1 = (cek1 * pow(commits[i],
pow(point, i, p - 1), p)) % p
        h = commit.other
          cek2 = (pow(g, share[0], p) * pow(h,
share[1], p)) % p
        return cek1 == cek2
```

```
def interpolate(points, mod, calc):
    res = 0
    for (x, y) in points:
        temp = y
        cnt = 0
        for (x1, y1) in points:
            if x1 == x:
                continue
            cnt += 1
             temp = (temp * (calc - x1) * pow(x -
x1, -1, mod)) % mod
        assert cnt == len(points) - 1
        res = (res + temp) % mod
    return res

def reconstruction(holders, threshold):
    mod = holders[0].prime
    for holder in holders:
        if holder.prime != mod:
            raise
           raise Exception("Invalid shares, cannot
reconstruct secret")
    if len(holders) < threshold:
       raise Exception("Not enough information to
reconstruct secret")
     points = [(holder.point, holder.share) for
holder in holders]
    return interpolate(points, mod, 0)
```

IV. Pengujian

Berikut ini diberikan kasus uji pada pennggunaan kedua algoritma tersebut. Pengujian dilakukan mulai dari perhitungan *share*, validasi tiap *share* dari tiap *participant* hingga rekonstruksi rahasia dari sejumlah *share* sebanyak *threshold.*

1) Nilai n/t konstan, yakni 3/4

p = 22801763489
s = 2345678910

(t, n)	Feldman (ms)	Pedersen (ms)
(3, 5)	29.98	91.01
(7, 10)	36.03	128.99
(18, 25)	40.03	74.68
(75, 100)	119.38	221.66
(375, 500)	2087.82	2178.85
(750, 1000)	8503.82	9201.83

2) Nilai n/t asimtot konstan, yakni 1, dan t = n - 1

p = 22801763489
s = 2345678910

(t, n)	Feldman (ms)	Pedersen (ms)
(4, 5)	32.99	82.01
(9, 10)	37.99	96.51
(24, 25)	41.00	65.99
(99, 100)	139.62	169.23
(499, 500)	2752.66	2868.20
(999, 1000)	11682.80	1270.54

3) Nilai t konstan dan kecil (t = 5)

p = 22801763489
s = 2345678910

(t, n)	Feldman (ms)	Pedersen (ms)
(4, 5)	28.99	58.01
(9, 10)	33.99	152.99
(24, 25)	37.00	59.98
(99, 100)	40.99	70.00
(499, 500)	66.01	94.00
(999, 1000)	94.99	149.99

IV. Analisis dan Kesimpulan

Algoritma Pedersen secara membutuhkan waktu yang lebih lama ketimbang algoritma Feldman. Hal ini diakibatkan karena pada algoritma Pedersen dibutuhkan lebih banyak komputasi, yakni dengan adanya polinomial tambahan pada algoritma tersebut. Akan tetapi, dari pengujian yang telah dilakukan, ketika nilai n bertambah, maka rasio waktu yang dibutuhkan untuk menjalankan algoritma Fieldman dan Pedersen semakin mendekati satu.

Pada aplikasinya, terutama pada *cloud computing*, jumlah server yang digunakan sebagai *participant* juga banyak, sehingga penggunaan algoritma Pedersen lebih praktis pada industri. Selain itu, terdapat kelebihan utama pada algoritma

Pedersen, yakni commit yang dibagikan kepada participant tidak memuat nilai dari g^s seperti yang dilakukan oleh algoritma Feldman, sehingga dalam teori informasi, algoritma ini dapat dikatakan lebih secure (aman) disbanding algoritma Feldman.

V. PRANALA

Kode yang digunakan dalam makalah ini dapat diakses pada pranala berikut: https://github.com/donbasta/verifiable-secret-sharing-lib

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa, beserta kedua orang tua yang telah mendukung penulis, serta Bapak Rinaldi Munir selaku pengampu mata kuliah IF4020 Kriptografi atas bantuan selama masa perkuliahan yang telah sangat membantu dalam pengerjaan tugas makalah mata kuliah Kriptografi. Penulis juga mengucapkan terima kasih kepada teman-teman dan pihak-pihak lain yang telah mendukung saya selama proses pengerjaan makalah ini.

REFERENSI

[1] Schoenmakers, B., 2020, *Lecture Notes Cryptographic Protocols*, Technical University of Eindhoven, diakses pada 21 Desember 2020, https://www.win.tue.nl/~berry/CryptographicProtocols/LectureNotes.pdf

[2] Feldman, P, 1987, *A Practical Scheme for Non-Interactive Verifiable Secret Sharing.* MIT, diakses pada 21 Desember 2020, http://www.cs.umd.edu/~gasarch/TOPICS/secretsharing/feldmanVSS.pdf

[3] Mortensen, M, 2007, *Secret Sharing and Secure Multi-party Computation*, University of Bergen, diakses pada 21 Desember 2020, http://www.ii.uib.no/~matthew/Masters/MichaelThesis.pdf

[4] Munir, R., 2020, *Slide Kuliah IF4020 Kriptografi: Kuliah Pengantar*.

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.

Jakarta, 21 Desember 2020

Farras Mohammad Hibban Faddila
13518017