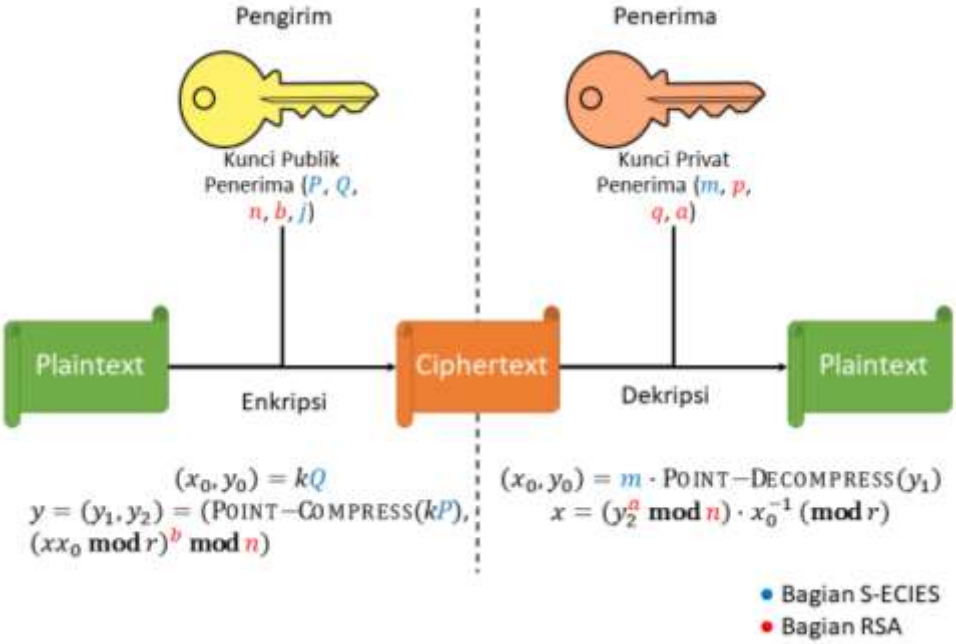
BAB IV
HASIL DAN PEMBAHASAN

4.1 Algoritma Kriptosistem Gabungan S-ECIES dan RSA



Gambar 1.1 Skema Kriptosistem Gabungan S-ECIES dan RSA

Algoritma kriptosistem gabungan S-ECIES dan RSA dapat dijelaskan sebagai berikut:

1. Pengirim menentukan *plaintext* yang akan dikirimkan kepada penerima. Penerima membangkitkan kunci publik dan kunci privat RSA dan S-ECIES yang akan digunakan.
2. Penerima mengirimkan kedua kunci publik kepada pengirim.
3. Pengirim melakukan enkripsi terhadap *plaintext* menggunakan kunci publik S-ECIES penerima. Hasil berupa *ciphertext* 1 yang berupa pasangan (y_1, y_s).
4. Pengirim kemudian melakukan enkripsi terhadap *ciphertext* 1 dengan cara mengenkripsi y_s menggunakan kunci publik RSA penerima menjadi y_2. Hasil berupa *ciphertext* 2 yang berupa pasangan (y_1, y_2) yang kemudian akan dikirimkan kepada penerima.

5. Penerima menerima *ciphertext* 2 dari pengirim, kemudian melakukan dekripsi terhadap y_3 menggunakan kunci privat RSA penerima. Hasil berupa *ciphertext* 1 yang berupa pasangan (y_1, y_s).
6. Penerima melakukan dekripsi terhadap *ciphertext* 1 menjadi *plaintext* menggunakan kunci privat S-ECIES penerima.

Kriptosistem gabungan ini dapat didefinisikan sebagai berikut.

Definisi 4.1.1 Kriptosistem Gabungan S-ECIES dan RSA

Diberikan E kurva eliptik pada lapangan $\mathbb{Z}_r$ ($r > 3$ bilangan prima) sedemikian sehingga terdapat $H = \langle P \rangle$ subgrup dari E dengan order j bilangan prima. Diberikan $n = pq$ dengan $n > r$, di mana p dan q merupakan bilangan prima. Diberikan $\mathcal{P} = \mathbb{Z}_r^*$ dan $\mathcal{C} = (\mathbb{Z}_r \times \mathbb{Z}_2) \times \mathbb{Z}_n$, didefinisikan:

$$\mathcal{K} = \{(n, p, q, a, b, E, P, m, Q, j) : Q = mP, ab \equiv 1 \pmod{\phi(n)}\}$$

Kunci publik dibentuk oleh P, Q, n, b dan j, dan kunci privat oleh $m \in \mathbb{Z}_j^*$, p, q dan a. Untuk $K = (n, p, q, a, b, E, P, m, Q, j)$, bilangan acak rahasia $k \in \mathbb{Z}_j^*$ dan *plaintext* $x \in \mathbb{Z}_r^*$, didefinisikan fungsi enkripsi

$$e_K(x, k) = (\text{POINT-COMPRESS}(kP), (xx_0 \bmod r)^b \bmod n)$$

di mana $kQ = (x_0, y_0)$ dan $x_0 \neq 0$.

Untuk *ciphertext* $y = (y_1, y_2)$ di mana $y_1 \in \mathbb{Z}_r \times \mathbb{Z}_2$ dan $y_2 \in \mathbb{Z}_n$, didefinisikan fungsi dekripsi

$$d_K(y) = (y_2^a \bmod n) \cdot x_0^{-1} \pmod r$$

di mana

$$(x_0, y_0) = m \cdot \text{POINT-DECOMPRESS}(y_1)$$

4.1.1 Proses Pembangkitan Kunci

Pada kriptosistem gabungan S-ECIES dan RSA ini, penerima melakukan pembangkitan kunci publik dan kunci privat yang kemudian dikirmikan kepada pengirim. Berikut langkah-langkah pembangkitan kunci publik dan kunci privat.

1. Pilih sebuah E kurva eliptik pada bilangan prima $\mathbb{Z}_r$ dengan $r > 127$, kemudian pilih sebuah $P \in E$ generator dari $H = \langle P \rangle \leq E$ dengan $|H| = j$ bilangan prima. Didapat nilai P, r dan j. Pada program akan

digunakan kurva eliptik P-256 dengan nilai-nilai P, r dan j diberikan dan r berukuran 256-bit.

2. Menggunakan Algoritma 2.4.1.1, penerima membangkitkan nilai-nilai n, b, p, q dan a dengan syarat $n > r$. Pada program akan digunakan $b = 65537$ dan n berukuran 2048-bit sehingga akibatnya syarat $n > r$ terpenuhi.
3. Penerima membangkitkan sebuah bilangan acak $m \in \mathbb{Z}_j^*$, kemudian menghitung $Q = mP$.
4. Didapat kunci publik adalah 5-*tuple* (P, Q, n, b, j) dan kunci privat adalah 4-*tuple* (m, p, q, a).

4.1.2 Enkripsi

Setelah menerima kunci publik, pengirim kemudian melakukan enkripsi terhadap *plaintext*. Langkah-langkah enkripsi adalah sebagai berikut.

1. Pengirim menentukan sebuah *plaintext* berupa teks alfanumerik. Pengirim kemudian mengubah setiap karakter pada *plaintext* menjadi bilangan bulat dengan aturan setiap karakter menjadi sebuah blok *plaintext* x_i dengan $i = 1, 2, 3, \ldots$. Pada program akan digunakan sistem ASCII untuk mengubah *plaintext* menjadi bilangan bulat.
2. Pengirim membangkitkan sebuah bilangan acak $k \in \mathbb{Z}_j^*$, kemudian menghitung nilai kP.
3. Pengirim kemudian melakukan kompresi titik kP menggunakan Definisi 2.4.3.1. Hasil kompresi menjadi nilai y_1.
4. Pengirim kemudian menghitung nilai $(x_0, y_0) = kQ$. Pengirim kemudian menghitung nilai $y_{s_i} = x_i x_0 \pmod r$ untuk setiap i.
5. Pengirim kemudian menghitung nilai $y_{2_i} = y_{s_i}^b \pmod n$ dan $y_2 = \{y_{2_i}\}$.
6. Didapat *ciphertext* adalah (y_1, y_2).

4.1.3 Dekripsi

Setelah menerima *ciphertext*, penerima kemudian melakukan dekripsi terhadap *ciphertext* untuk memperoleh *plaintext* kembali. Langkah-langkah dekripsi adalah sebagai berikut.

1. Penerima menghitung nilai $(x_0, y_0) = m \cdot \text{POINT-DECOMPRESS}(y_1)$ menggunakan Algoritma 2.4.3.1.
2. Penerima kemudian menghitung nilai $y_{s_i} = y_{2_i}^a \pmod n$ untuk setiap $y_{2_i} \in y_2$.
3. Penerima kemudian mendapatkan kembali blok-blok *plaintext* dengan menghitung $x_i = y_{s_i} \cdot x_0^{-1} \pmod r$.
4. Penerima kemudian mengubah kembali setiap blok x_i pada *plaintext* menjadi karakter alfanumerik.

4.1.4 Contoh Enkripsi dan Dekripsi

Contoh pembangkitan kunci, enkripsi dan dekripsi pada kriptosistem gabungan S-ECIES dan RSA adalah sebagai berikut: Misalkan Alice ingin mengirimkan *plaintext* berupa bilangan bulat $x = 9$. Bob kemudian memilih kurva eliptik E yang didefinisikan sebagai himpunan (x, y) yang memenuhi $y^2 = x^3 + x + 6$ pada $\mathbb{Z}_{11}$. Bob kemudian membangkit membangkitkan dua buah bilangan prima $p = 7$ dan $q = 13$, kemudian menghitung nilai $n = pq = 7 \cdot 13 = 91$ dan nilai $\phi(n) = (p-1)(q-1) = 6 \cdot 12 = 72$. Bob kemudian memilih suatu bilangan $b = 5$ yang relatif prima terhadap $\phi(n)$. Bob kemudian menghitung nilai $a = 29$ menggunakan algoritma Euclid yang diperluas.

Karena $|E| = 13$ merupakan bilangan prima, maka berdasarkan Teorema 2.2.1, E merupakan grup siklik dan seluruh titik pada E kecuali $\boldsymbol{\mathcal{O}}$ merupakan generator dari E. Bob kemudian mengambil sebuah generator $P = (2, 7) \in E$ dan sebuah bilangan bulat acak $m = 7 \in \mathbb{Z}_{13}$. Bob kemudian menghitung $Q = mP = 7(2, 7) = \underbrace{P \oplus P \oplus P \oplus \ldots \oplus P}_{7} = (7, 2)$. Maka diperoleh kunci publik Bob adalah $\big((2,7), (7,2), 91,5,13\big)$ dan kunci privat Bob adalah $(7,7,13,29)$. Bob kemudian mengirimkan kunci publiknya kepada Alice. Alice kemudian memilih

sebuah bilangan bulat acak $k = 6 \in \mathbb{Z}_{13}$. Alice kemudian menghitung nilai $(x_0, y_0) = kQ = 6(7,2) = (8,3)$. Alice kemudian mengenkripsi *plaintext* menggunakan kunci publik Bob. Hasil enkripsi adalah:

$$\begin{aligned} y = (y_1, y_2) &= (\text{POINT-COMPRESS}(kP), (xx_0 \bmod r)^b \bmod n) \\ &= (\text{POINT-COMPRESS}\big(6(2,7)\big), (9 \cdot 8 \bmod 11)^5 \bmod 91) \\ &= ((\text{POINT-COMPRESS}(7,9), 41) \\ &= ((7,1), 41) \end{aligned}$$

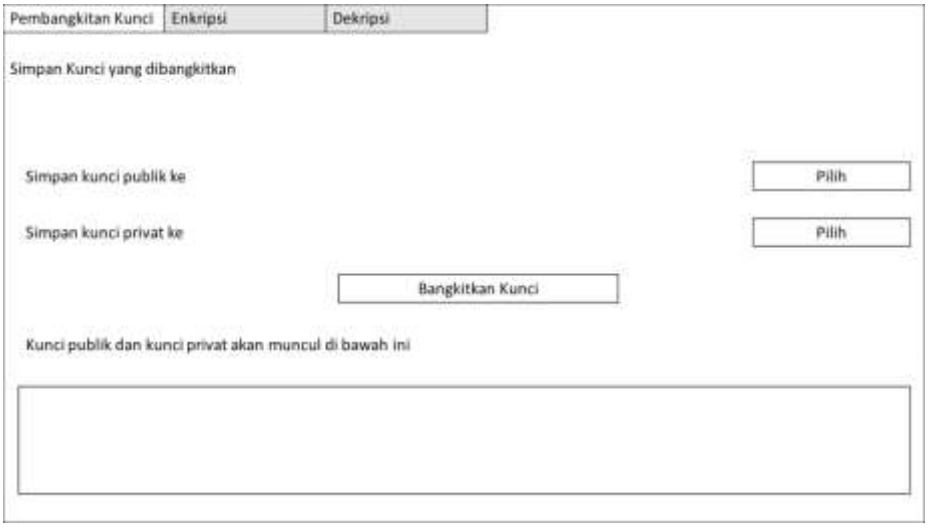
Alice kemudian mengirimkan *ciphertext* kepada Bob. Bob kemudian mendekripsi *ciphertext* menggunakan kunci privat Bob. Bob kemudian menghitung $(x_0, y_0) = m \cdot \text{POINT-DECOMPRESS}\big((7,1)\big) = 7 \cdot (7,9) = (8,3)$. Hasil dekripsi adalah:

$$\begin{aligned} x &= (y_2^a \bmod n) \cdot x_0^{-1} \pmod r \\ &= (41^{29} \bmod 91) \cdot 8^{-1} \pmod{11} \\ &= 9 \end{aligned}$$

Maka Bob mendapatkan pesan asli adalah $x = 9$.

4.2 Perancangan Program Komputer

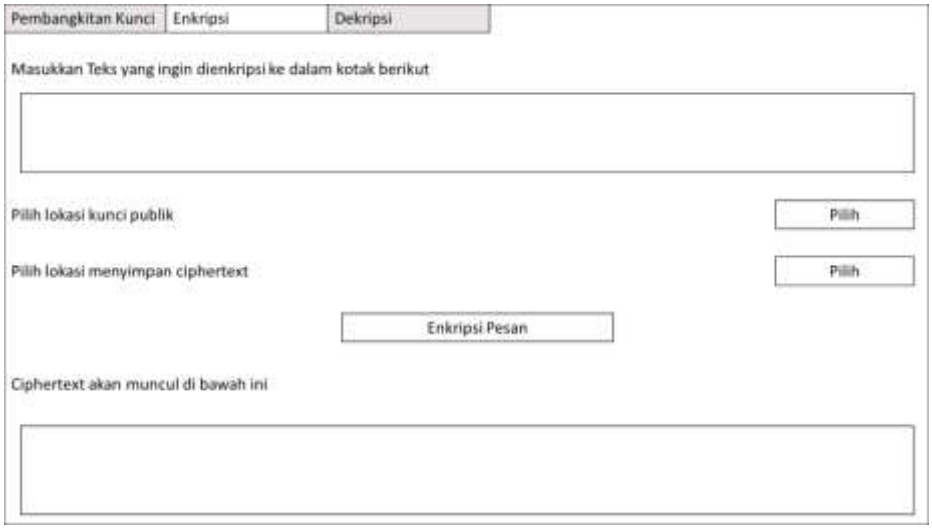
Untuk melakukan validasi dan mempermudah melakukan pembangkitan kunci, enkripsi dan dekripsi pada kriptosistem gabungan, dilakukan pembuatan sebuah program komputer. Program dibuat menggunakan bahasa pemrograman Python 3.5. Rancangan tampilan awal program adalah berikut ini.



Gambar 1.2 Rancangan Tampilan Awal Program

Pada tampilan awal, pengguna berada pada tampilan untuk melakukan pembangkitan kunci. Pengguna dapat beralih ke tampilan untuk melakukan enkripsi dan dekripsi menggunakan *tab* yang ada. Pada tampilan pembangkitan kunci, terdapat dua buah tombol yang masing-masing berfungsi untuk memilih lokasi direktori penyimpanan kunci publik dan kunci privat. Terdapat pula tombol yang berfungsi untuk memulai proses pembangkitan kunci. Apabila kunci selesai dibangkitkan, kunci publik dan kunci privat akan disimpan sebagai *file object* Python pada lokasi direktori yang sudah dipilih. Kunci publik dan kunci privat juga akan dicetak ke dalam text box yang terdapat pada bagian bawah tampilan.

Tampilan *tab* enkripsi dapat dilihat pada gambar berikut ini.



Gambar 1.3 Rancangan Tampilan *Tab* Enkripsi

Pada *tab* enkripsi, pengguna dapat melakukan enkripsi terhadap *plaintext*. Terdapat sebuah *text box* untuk memasukkan isi *plaintext* yang akan dienkripsi. Terdapat pula dua buah tombol yang masing-masing berfungsi untuk memilih lokasi kunci publik yang akan digunakan (berupa *file object* Python) dan lokasi untuk menyimpan *ciphertext* sebagai *file object* Python. Terdapat pula sebuah tombol untuk memulai proses enkripsi. *Ciphertext* yang dihasilkan kemudian akan disimpan sebagai *file object* Python pada lokasi yang sudah dipilih. *Ciphertext* yang dihasilkan kemudian akan disimpan sebagai *file object* Python pada lokasi yang sudah dipilih dan juga dicetak ke dalam *text box* yang terdapat pada bagian bawah.

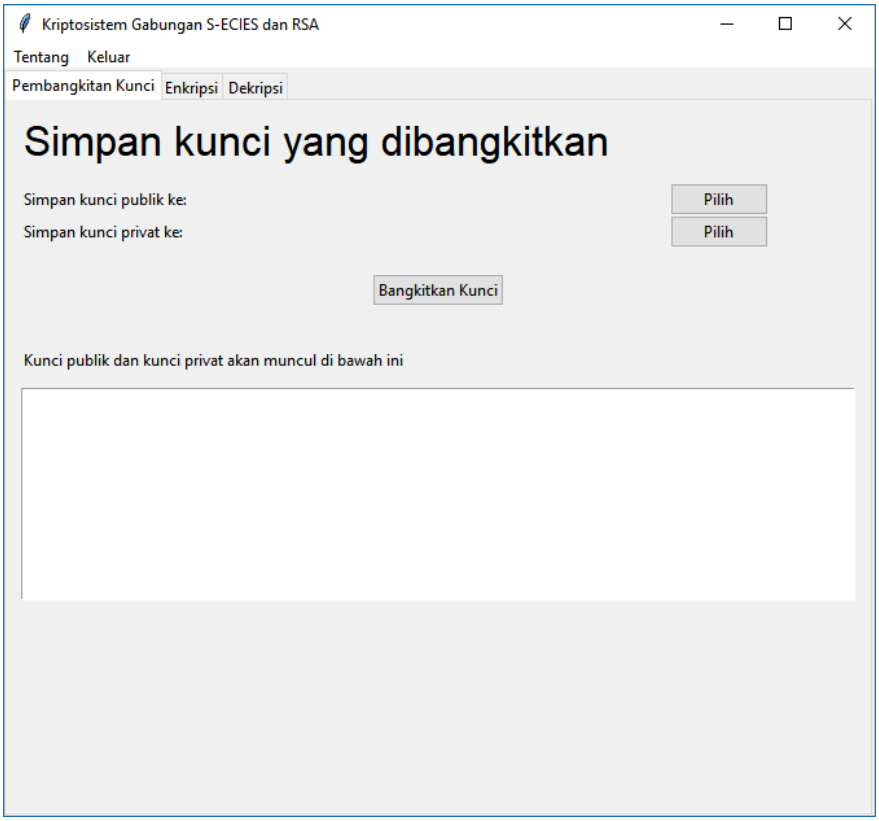
Tampilan *tab* dekripsi dapat dilihat pada gambar di bawah ini.

Gambar 1.4 Rancangan Tampilan *Tab* Dekripsi

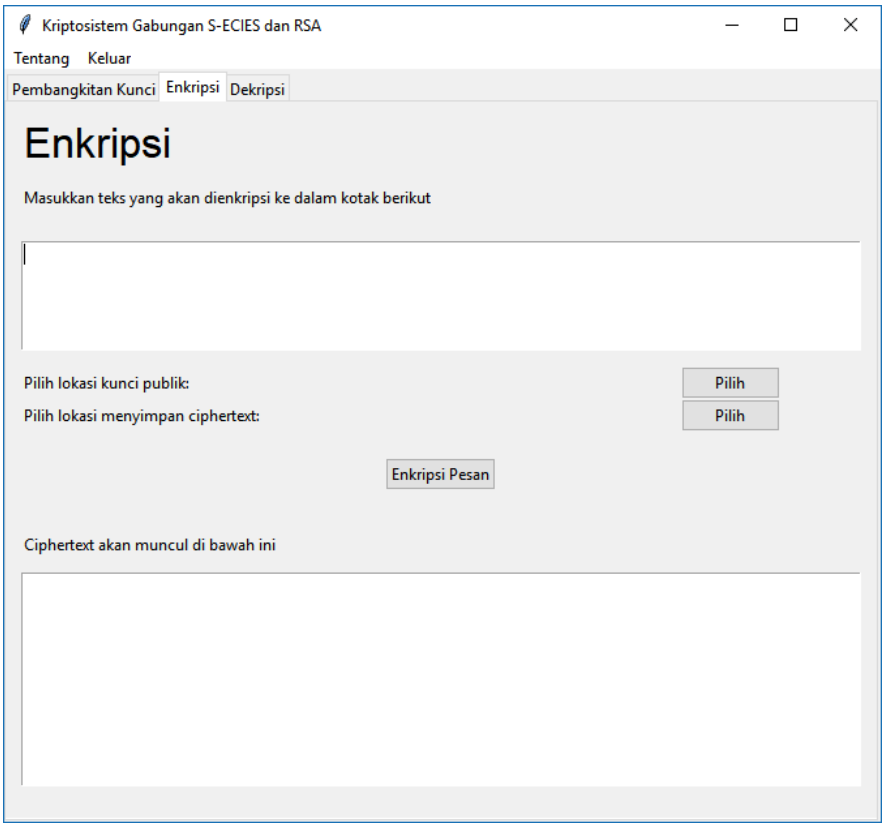
Pada *tab* dekripsi, pengguna dapat melakukan dekripsi terhadap *ciphertext*. Terdapat dua buah tombol yang masing-masing berfungsi untuk memilih lokasi kunci privat yang akan digunakan dan lokasi *file ciphertext* yang akan didekripsi (berupa *file object* Python). Terdapat pula sebuah tombol untuk memulai proses dekripsi. *Plaintext* yang dihasilkan kemudian akan dicetak ke dalam *text box* yang terdapat pada bagian bawah.

4.3 Pembuatan Program Komputer

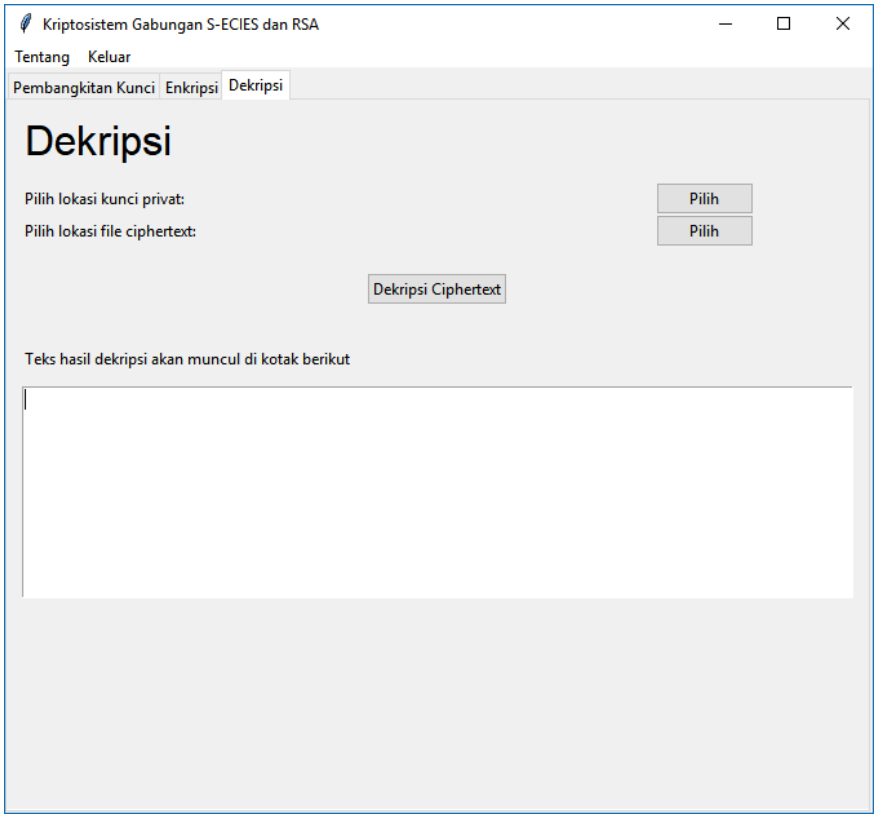
Setelah dilakukan perancangan tampilan program, maka dilakukan pembuatan program. Program dibuat menggunakan bahasa pemrograman Python versi 3.5. Pada program digunakan kurva eliptik P-256 untuk parameter-parameter S-ECIES, sedangkan untuk paramater RSA dibangkitkan modulus berukuran 2048-bit. Kunci publik dan kunci privat yang sudah dibangkitkan masing-masing disimpan berupa *file object* Python. *Plaintext* berupa teks alfanumerik yang kemudian akan diubah menjadi bilangan bulat menggunakan sistem ASCII. *Ciphertext* hasil enkripsi disimpan berupa *file object* Python. Berikut tampilan program yang sudah selesai dibuat.



Gambar 1.5 Tampilan Awal Program



Gambar 1.6 Tampilan *Tab* Enkripsi



Gambar 1.7 Tampilan Tab Dekripsi

4.4 Validasi Program dengan Contoh

Setelah program selesai dikonstruksi, maka diperlukan validasi untuk mengecek apakah *plaintext* hasil enkripsi dan dekripsi masih sama dengan *plaintext* aslinya atau tidak. Validasi dilakukan menggunakan beberapa contoh berikut yang didapatkan dari *output* aplikasi.

Contoh 4.4.1 Pembangkitan Kunci, Enkripsi dan Dekripsi

Misalkan Bob ingin membangkitkan kunci publik dan kunci privat. Bob kemudian secara acak menggunakan program yang sudah dibuat membangkitkan nilai-nilai

$$p = 920643522497353418737712195642577605881083301$$
$$086761296152544510880021732742284684652155585 3$$
$$466839409027585120939270419249024622038172369 2$$
$$927276084443932387732568583132020131421828676 2$$
$$101031109993952783128149127312416583655909402 6$$
$$548512132274144100119410169758696256761831018 0$$

$$3242183367649224169848359077183 93$$

dan

$$q = 10638550517196960346874485536990864717487754 7$$
$$3109734134906906358000523496280407609559045949$$
$$5610539789261945636330364686219251202237630757$$
$$9032757616276010804677540558934092806177464899$$
$$3502992658597944236340658392158972378136423367$$
$$6152421365432294971423615028962222256237430433$$
$$4143016783584714284934593655129031$$

Selanjutnya menggunakan program, Bob menghitung nilai $n = p \cdot q$ dan $\phi(n) = (p-1)(q-1)$. Bob kemudian mendapatkan nilai

$$n = 9794312622418250611582632524937671160353554 92$$
$$5961788632399390713234236837654875507147867891$$
$$9065869570123784126659795252083056967086601361$$
$$7248276765339614972676608256208954640868989409$$
$$8261472898029922067868944835635127741056474176$$
$$0278853134058036758515724201498169144708426993$$
$$3539068082064527129616863457867258739955303591$$
$$1014185050940764811597272676579335359122807896$$
$$3483391107829615142283188927261998642040272608$$
$$5179800654762262310789436196828064778716940080$$
$$9196841759645489283120949215437591076622222932$$
$$7021704804727841725130319107517817554645219699$$
$$1623833951873092689889131133543313712117581158$$
$$1357986605226967183$$

dan

$$\phi(n) = 9794312622418250611582632524937671160353554 92$$
$$5961788632399390713234236837654875507147867891$$
$$9065869570123784126659795252083056967086601361$$
$$7248276765339614972676608256208954640868989409$$
$$8261472898029922067868944835635127741056474176$$
$$0278853134058036758515724201498169144708426993$$

353906808206452712961686345786723889496956142060688425348658306451895096907019157095797019512614510562600592449988977725727090904120443655442253101965679403496516039370086877534686808660044316505034231701833499218620306742453637430002235997208370336168526774747547461295449055255229092667437177137613187209186698860554619446560671756641197 60

Menggunakan program, Bob kemudian memilih $b = 65537$ yang relatif prima terhadap $\phi(n)$. Menggunakan program, Bob kemudian menghitung nilai a. Bob kemudian mendapat

$a = 848052343060784059698899957672937558425656999$
$601183541715574141955216142926230321083682364762090396970603514663691768218671155325128748778924435255290321364107402110362594260505541496406526910099037120501840401060394828560951345946378961494309680622966154315610282174286952128547354279030679715233204659659952978225166430775870940817395403805367833298092541397698654082857472326370328084452222412394449005175503253387566492842889273359064218700115957792291017932293938446171248630886941217911702764434458336955747729921446373227218267516720057169469473658121380671435990944199142678564538129763711356109049773770755 3$

Menggunakan kurva eliptik P-256, Bob mendapatkan nilai-nilai P, r dan j. Bob kemudian secara acak menggunakan program memilih

$m = 942214518246712087287706728002094215814333720$
$57210513923530592929731245932370$

dan menghitung nilai

$Q = mP = (105505568987383554617606108621433258629663897645614308758532755889336305288 97, 146780678250$

$$75905079268181434868813792172522895420767577194763811414821050238)$$

Bob kemudian mendapatkan kunci publik (P, Q, n, b, j) dan kunci privat (m, p, q, a). Bob kemudian mengirimkan kunci publik kepada Alice. Misalkan Alice ingin mengirimkan *plaintext* "`Si`". Menggunakan program, *plaintext* kemudian direpresentasikan ke dalam bilangan bulat dengan aturan setiap karakter menjadi sebuah blok *plaintext* menggunakan sistem ASCII. Hasilnya dapat dilihat pada tabel berikut.

Tabel 1.1 Representasi *Plaintext* Dalam Sistem ASCII

Karakter	Kode Desimal ASCII
S	83
i	105

Menggunakan program, Alice kemudian memilih secara acak nilai $k \in \mathbb{Z}_j^*$, didapat

$$k = 37495901854362262360071160586012906099495390560561928220849713686598938037 91$$

Menggunakan program, Alice kemudian menghitung nilai

$$kP = (54365123126044252483440346801968900933866645708188482675193928330480957932190, 998009392133982877372052297270255763512426152866852825182974395870298235829 34)$$

dan

$$(x_0, y_0) = kQ = (10465785949653054449385251479884374542634000728600491786429317632245442262115 1, 1127403495488240214509714939587005782739315545666431588267865372904158704620 75)$$

Alice kemudian melakukan kompresi titik pada kP menggunakan program, hasil kompresi menjadi nilai y_1. Didapat

$$y_1 = (54365123126044252483440346801968900933866645708188482675193928330480957932190, 0)$$

Alice kemudian menghitung nilai $y_{3_i} = \left(x_i x_0 \ (\mathbf{mod}\, r)\right)^b \ (\mathbf{mod}\, n)$ untuk setiap i. Didapat

$y_{3_1} = \left(83 \cdot x_0 \ (\mathbf{mod}\, r)\right)^b \ (\mathbf{mod}\, n) =$ 55124996454178220643281186634664726754011298479739452462802836216878663260231023633542916095939175859575650409830175722740672431243809176618383118865623830542679092156464718372739743939590086749023959614033433724927928666939224125790983080430831446993123859210631307187365625721547736019538584394119212741017572753260658886302921879695543594524820199424836981630295894841445567606040910572876546787063692801372770691093535342390678555318238639616302884657811370632731916989193853459762486399580737433506798471103619319430408303536805265612285659345012680834305853035215097669656185172640913766732128386859260294377 28

dan

$y_{3_2} = \left(105 \cdot x_0 \ (\mathbf{mod}\, r)\right)^b \ (\mathbf{mod}\, n) =$ 211307409388948178982779095976372964829248341621627196208365115110012613699798668621300023963621681111034036215702982425793733688453915702348025355230523233981354734505427162391128454282200298044112392636885715201106164418290137650064117372268831567877951799128800293751896370598120576200718625734819751312642662924914926602678921167365167419645592628526768979168814318329302810550053555473617196928587324452572902907541413617033510292381829307250814911189362368327185598983012407889770368663160688710173815839079126489037753186683245773320148461643159304067521033902906056498709955009401205684580369488622998588819 4

Alice kemudian mendapat $y_3 = \{y_{3_i}\}$. *Ciphertext* yang diperoleh adalah (y_1, y_3). Alice lalu mengirimkan *ciphertext* kepada Bob. Bob kemudian melakukan dekripsi *ciphertext* yang diterimanya dari Alice. Bob kemudian menghitung nilai

$$(x_0, y_0) = m \cdot \text{POINT-DECOMPRESS}(y_1) = (044252483440$$
$$346801968900933866645708188482675193928330480$$
$$957932190, 998009392133982877372052297270255763$$
$$512426152866852825182974395870298235829 34)$$

Bob kemudian mencari invers multiplikatif x_0. Didapat

$$x_0^{-1} = 156042874372269530193038166798050028699245815$$
$$080565886526677597140829396032 74$$

Bob kemudian mencari nilai-nilai $x_i = \left(y_{3_i}^a \bmod n\right) \cdot x_0^{-1} \pmod r$. Didapat

$$x_1 = \left(y_{3_1}^a \bmod n\right) \cdot x_0^{-1} \pmod r = 83$$
$$x_2 = \left(y_{3_2}^a \bmod n\right) \cdot x_0^{-1} \pmod r = 105$$

Bob kemudian mengembalikan setiap blok *plaintext* hasil dekripsi menjadi teks alfanumerik menggunakan sistem ASCII menggunakan program. Didapat *plaintext* hasil dekripsi adalah "`Si`". Karena *plaintext* hasil dekripsi dari *ciphertext* sama dengan *plaintext* asli, maka program berjalan dengan baik (program valid).