Rīgas 64. Vidusskola

Steganogrāfijas un kriptogrāfijas pielietošana datu apslēpšanā

Pētnieciskais darbs Programmēšanā.

Darba autors:
Rīgas 64. vidusskolas 12.DIT klases skolēns
Ernests Pēteris Kungs

Darba vadītājs:
Rīgas 64.vidusskolas programmēšanas skolotājs
Edvards Bukovskis

Rīga 2024

Abstract

Application of steganography and cryptography in data hiding. Research paper by Ernests Pēteris Kungs, supervisor Edvards Bukovskis, programming teacher of Riga 64th secondary school.

The work examines the application of steganography methods for the purpose of hiding data in pictures and the use of cryptographic methods for data encryption, as well as the development of software, combining both methods to create a more complex data hiding method.

The developed software of the work would use a method that would allow hiding data with steganography, so that third persons would not suspect the hidden data is being sent. In case of the data being acessed, the data would be encrypted, denying easy acess to the data.

Anotācija

Steganogrāfijas un kriptogrāfijas pielietošana datu apslēpšanā. Ernesta Pētera Kunga zinātniski pētnieciskais darbs, darba vadītājs Rīgas 64.vidusskolas programmēšanas skolotājs Edvards Bukovskis.

Darbā apskatīts steganogrāfijas metožu pielietojums datu apslēpšanai bildēs un kriptogrāfijas metožu pielietošana datu šifrēšanā, kā arī programmatūras izveide, apvienojot abas metodes, kompleksākas datu apslēpšanas metodei.

Darba rezultātā izstrādātā programmatūra pielietotu metodi, kura apslēptu datus ar steganogrāfiju, lai nepiederošās personas nenojaustu, ka tiek nosūtīti slēpti dati. Datu piekļuves gadījumā, iegūtie dati būtu šifrēti, kas liegtu vieglu piekļuvi nodotajai informācijai.

Satura rādītājs

Ievads

Mūsdienu sabiedrībā cilvēce ir pakļauta lielam datu apjomam internetā, kas ietver gan privātu informāciju, gan publiski pieejamu informāciju. Šis lielais datu apjoms izraisa drošības riskus, tostarp konfidenciālu datu noplūdi nepiederošām personām. Lai liegtu ļaunprātīgu piekļuvi informācijai, ikdienā pielietotas dažādas datu apstrādes metodes. Viena no šīm metodēm ir kriptogrāfija, kuras galvenais mērķis ir datu iešifrēšana, lai ierobežotu nepiederošas personas piekļuvi šiem datiem. Šī metode jau ir novērota praksē, no senās Ēģiptes laikiem līdz mūsdienām, kur tā tiek pielietota dažādās nozarēs, kā, piemēram, datorzinātnēs, militārajā jomā un datu pārraidē. Steganogrāfija ir metode, kura apstrādā konkrētu informāciju kādā priekšmetā vai datnē, lai izvairītos no šīs informācijas atklāšanas. Šo metožu apvienojums ļauj nodot informāciju starp cilvēkiem, citiem nemanot, un ja gadās risks, ka šī informācija nonāk nepiederošas personas rokās, tad tā tiek šifrēta, nepieļaujot datu noplūdi. Šajā darbā izpētītas abas metodes, izstrādāta programmatūra, kura piedāvā datu šifrēšanu un apslēpšanu.

Darba mērķis: Izveidot jaunu metodi, kā iešifrēt datus, apslēpjot to saturu un mazinot to redzamību.

Hipotēze: Darba autoram izdosies gan izpētīt dažādas kriptogrāfijas un steganogrāfijas metodes datu šifrēšanā un apslēpšanā, gan apvienot šīs abas metodes, izveidojot metodi, kura ir efektīvāka, lai nodotu privātu informāciju.

Pētāmā problēma: Kriptogrāfrijas un steganogrāfijas metožu apvienošana jaunas metodes izveidei datu apslēpšanai.

Darba uzdevumi:

1. Izpētīt dažādas kriptogrāfijas metodes datu iešifrēšanā.

2. Izpētīt dažādas steganogrāfijas metodes datu apslēpšanai bildēs.

3. Izpētīt, kā šīs abas metodes var realizēt, rakstot kodu.

4. Veikt šo abu metožu apvienojuma programmatūras izstrādi.

5. Izveidot metodi, kuras rezultātā iešifrēts saturs ar izvēlētu kriptogrāfijas metodi, kā arī dati tiek apslēpti bildē ar steganogrāfijas palīdzību.

Izmantotās darba metodes: Literatūras analīze, lai izpētītu dažādu kriptogrāfijas un steganogrāfijas metožu pielietojumu datu aizsardzībā.

1. Literatūras apskats

1.1 Kriptogrāfijas principi

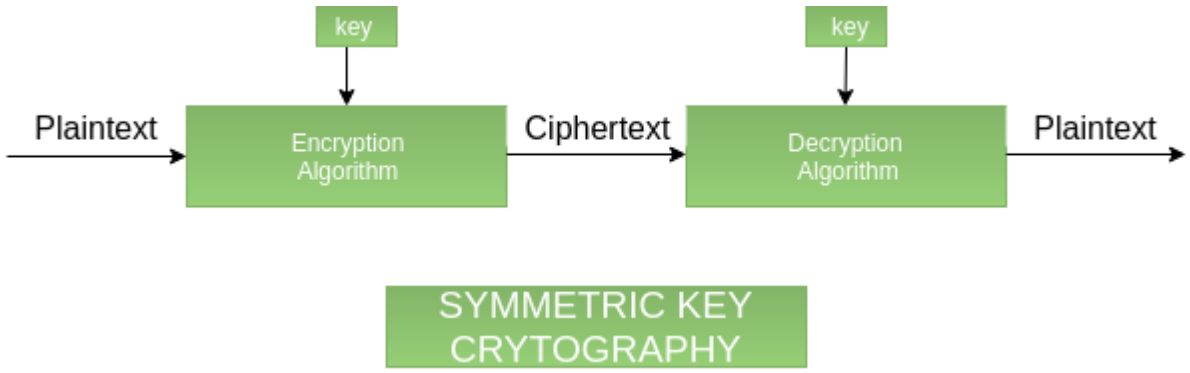
Šīs metodes pielietojuma mērķis ir panākt to, ka var komunicēt 2 personas slepeni, novēršot trešās personas piekļuvi koplietotajiem datiem. Kopumā, kriptogrāfija ir zinātnes nozare, kas attīstījusies no kriptoloģijas. Tajā pētītas dažādas metodes teksta iešifrēšanai un apslēpšanai. Turpretim, kriptoanalīze ir zinātne, kura pēta dažādas metodes, kā atšifrēt kriptogrāfijas rezultātā izveidoto tekstu. [1]

Kriptogrāfiju var iedalīt tādās klasēs, kā:

- Simetriskā kriptogrāfija;
- Asimetriskā kriptogrāfija;
- Jaukšanas metode.

1.2 Simetriskā kriptogrāfija

Simetriskā kriptogrāfija ir metode, kura iešifrē tekstu ar vienu atslēgu, ar kuru ikviens var atšifrēt doto tekstu, kuram ir piekļuve dotajai atslēgai. Šī metode ir nedrošāka, jo, lai doto tekstu spētu atšifrēt konkrētā persona, tai vajag saņemt šo slēpto atslēgu, kas varētu izraisīt vairākas ar drošību saistītas problēmas, jo šīs atslēgas nodošanai ir jāveido kontakts. Ja kāds piekļūst pie šīs apmaiņas, tad teksts ir pakļauts atšifrēšanas riskam. [1]

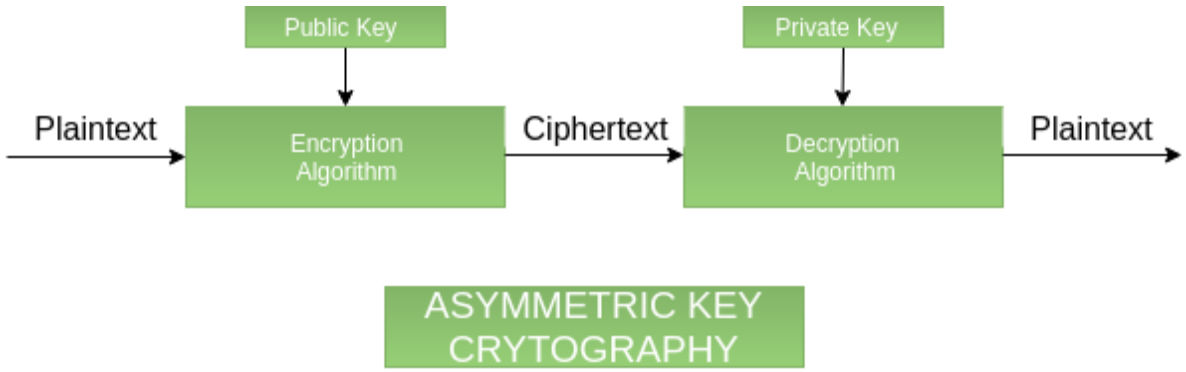


(1. Attēls. Simetriskās kriptogrāfijas process diagrammā. [1])

Tomēr, šai metodei ir ieguvumi. Process, kas ir nepieciešams datu apstrādei, ir manāmi atvieglots, nav pārāk garš un komplekss. Atslēgām nevajag būt pārāk sarežģītām teksta iešifrēšanai, un šī metode ļauj veidot divvirzienu komunikāciju. Abas personas var būt gan sūtītāji, gan saņēmēji, kā arī droša algoritma izveide neaizņem tik daudz laika salīdzinājumā ar asimetrisko kriptogrāfiju. Atsevišķos gadījumos nav nekādas vajadzības veidot asimetriskās iešifrēšanas metodes, ja ir sarakste starp cilvēkiem, kuriem ir jau zināma iešifrēšanas atslēga vieglas informācijas pārraidei un atšifrēšanai. [2]

1.3 Asimetriskā kriptogrāfija

Asimetriskā kriptogrāfija ir metode, kura iešifrē tekstu, izmantojot 2 atslēgas. Pirmā ir publiski pieejama, bet otrā ir privāta. Publiski pieejamo atslēgu izmanto, doto datu iešifrēšanā, bet privāto izmanto datu atšifrēšanā. Abas šīs atslēgas atšķiras viena no otras, un pat ja kāds uzzina šo publisko atslēgu, viņi nav spējīgi atšifrēt doto tekstu, jo tas paredzēts ir tikai privātās atslēgas glabātājam (skatīt 2. attēlu).[1]



(2. Attēls. Asimetriskās kriptogrāfijas pielietojums datu iešifrēšanā un atšifrēšanā, izmantojot publisko un privāto atslēgu [1.])

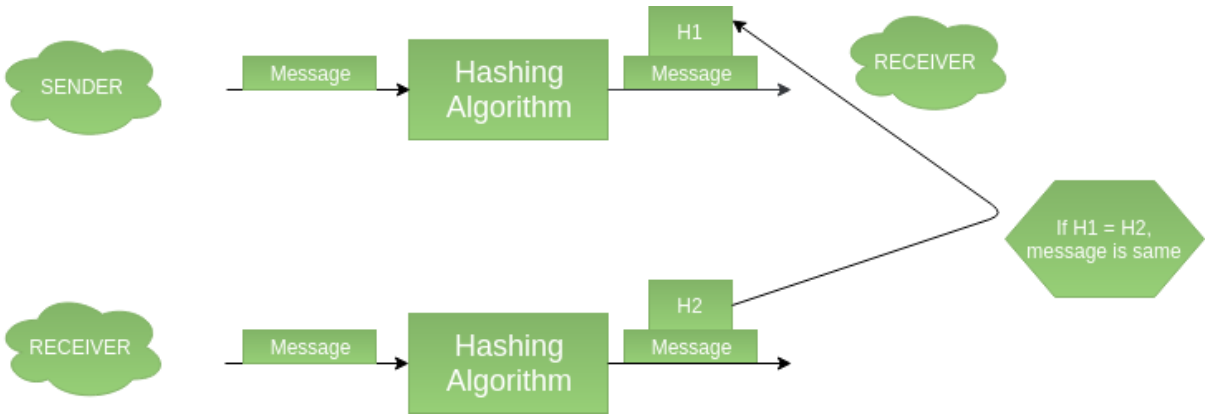
Šī metode balstās uz atslēgu veidošanu, kuras ir saistītas ar matemātisku algoritmu savā starpā bet ir atšķirīgas pēc būtības. Ir sarežģīti uzlaust algoritmu, izmantojot tikai vienu atslēgu, bet viegli, izmantojot abas. Šī metode ir ērta ātrai savienošanai un atslēgas nodošanai, jo publiskā atslēga tiek uzreiz nodota konkrētajam sūtītājam, tomēr tā nedod iespēju atšifrēt datus, ja šai atslēgai piekļūst kāds cits. Šī metode ir izplatīta digitālajā vidē, veidojot savienojumus starp datoriem un mājaslapām, izmantojot SSL/TLS metodi. [3]

Tomēr, asimetriskās kriptogrāfijas metodei ir trūkumi. Šāda iešifrēšanas metode pieļauj tikai vienvirziena komunikāciju, jo tikai saņēmējs var atšifrēt tekstu ar sev pieejamo privāto atslēgu. Šīm atslēgām jābūt pēc iespējas garākām, lai apgrūtinātu ļaunprātīgu piekļuvi un atslēgas uzlaušanu. Šī īpašība rezultējas ar ilgāku atšifrēšanas laiku, kā arī izveidot šādu algoritmu ir sarežģīti. Tas jāizveido pēc iespējas kompleksāks, kas pieprasa augstu lietpratību matemātikā, algoritmu veidošanā. [3]

1.4 Jaukšanas (*Hashing*) metode

Jaukšanas metode ir vēl viena metode, kura tiek pielietota šifrēšanā. Tā ir funkcija, kura pārvērš doto tekstu par noteiktu "hash value" ar noteiktu garumu. Šis process pārbauda, vai ziņa nav bijusi mainīta sūtīšanas laikā, jo abiem "hash values" ir jābūt identieskiem.[1]

Šī metode ir vairāk pielietota nevis teksta iešifrēšanai, bet gan kā papildus metode, kura tiek pielietota, lai nodrošinātu, ka iešifrētais teksts nav ticis mainīts sūtīšanas laikā un ka informācija, kas tiek sūtīta, un ir autentiska konkrētajam sūtītājam. Mūsdienās ir pieejami daudz dažādi *hashing* algoritmi, bet visbiežāk sastopamais ir SHA256("Secure hashing algorithm"), kuru izstrādāja CIA(Centrālā izmeklēšanas aģentūra). *Hashing* metode arī ir laba ar to, ka vienmēr tiks izveidoti vienādi *hash values*, bet tos atšifrēt aizņemtu ļoti ilgi, jo vienīgā metode kā to var panākt ir testējot katru "character" pēc kārtas, līdz kamēr sakrīt *hash values*. Tāpēc ir diezko viegli izveidot *hash value*, bet ne atšifrēt to, kas ļauj droši un apslēpti nosūtīt datus, un pārliecināties, ka teksts nav bijis mainīts. [4]



(3. Attēls. Asimetriskās kriptogrāfijas pielietojums datu iešifrēšanā un atšifrēšanā, izmantojot publisko un privāto atslēgu [1.])

1.5 Steganogrāfijas principi

Steganogrāfija ir metode, ar kuras palīdzību jebkurā vietā var apslēpt kādu informāciju tā, lai to nespētu no ārpuses pamanīt nepiederoša persona, bet tā, lai tai varētu piekļūt cilvēks, kam šī informācija nosūtīta. Ar šo metodi var apslēpt jebko, kas varētu iekļaut tekstu, bildi, video vai audio datus. [5]

Steganogrāfija bija populārākā pielietošanas metode senajā Grieķijā, kur cilvēki uzrakstīja tekstu, iegrebjot to koka plāksnē, pēc tam pārklājot to pašu plāksni ar vasku, lai nespētu to citi pamanīt. [6]

Ir dažādas metodes, kā pielietot steganogrāfiju - gan digitāli, gan dabā. Šajā darbā apskatīta bilžu steganogrāfija, kurā dati apslēpti bilžu failos. Dažas metodes bilžu steganogrāfijai būtu:

1. Mazāk svarīgāko bitu jeb *Least significant bit* (LSB)

2. Pikseļu vērtību atšķiršana jeb *Pixel value differencing* (PVD)

3. Malu balstīta datu apslēpšanas metode jeb *Edges based data embedding method* (EBE)

4. Nejaušas pikseļu mainīšanas metode jeb *Random pixel embedding method* (RPE) [7]

1.6 LSB(“Least Significant Bit”) steganogrāfijas metode

LSB metodes mērķis ir apslēpt informāciju bildē, pamainot krāsu bitus tā, lai mainītajai bildei nebūtu manāmas atšķirības. Šī metode pielieto bildes skenēšanu, kuras rezultātā iegūtas bildes pikseļu RGB krāsu vērtības. Pēc tā seko teksta pārveidošana par bināro tekstu, kur katrs burts apzīmēts ar 0 un 1, un ir 8 bitu formātā. Ar iegūto tekstu mainās RGB vērtības, pievienojot tām 1 vai nemainot tās, lai tās atbilstu tekstam, kur 0 reprezentē RGB vērtības ar pāra skaitli, bet 1 tiek reprezentēts ar nepāra skaitli. Piemēram, ja būtu 8 biti kā “001100111”, tad būtu nepieciešami 3 pikseļi bildē, jo katram pikselim ir 3 RGB vērtības. Rezultātā, ja pirmajam pikselim RGB vērtības ir, piemēram, 31; 44 un 32, tad LSB metode, vadoties pēc dotā 8 bitu teksta, ņemtu pirmos 3 bitus, kas būtu “001” un pamainītu šīs vērtības uz 32; 44 un 33, lai izpildītos dotie algoritma nosacījumi. Šis process atkārtotos atlikušajiem pikseļiem, lai visi biti dotajā tekstā būtu izmantoti. [8]

LSB metodei ir 2 trūkumi:

1. Nepareiza lieluma bildes izvēles gadījumā fiziski varētu nepietikt vietas tekstam dotajā bildē, jo bitu daudzums pārsniegtu pieejamās RGB vērtības. Bildes pieejamo kapacitāti var noskaidrot ar formulu:
$$Attēla\ kapacitāte = Attēla\ garums * Attēla\ platums * 3,$$
kur attēla garums un platums ir mērīts pikseļos.
2. Sliktas kvalitātes attēla izvēles gadījumā metode var neizpildīties, jo ir divas "image compression" jeb attēla saspiešanas metodes, kuras ir sastopamas attēlu failos:
    1. "Lossy compression" metode
    2. "Lossless compression" metode

No šīm 2 metodēm, ja tiek izvēlēta "lossy compression" metode, tad tā var traucēt LSB steganogrāfijas metodei, jo tā var nekorekti saglabāt pikseļu RGB vērtības, lai mazinātu faila lielumu. Šis negatīvi ietekmētu šo metodi, jo tad nevar paļauties, ka konkrētajai bildei ir korekti saglabātas vērtības. "Lossy compression" metodes attēla izmantošana ietver risku, ka saglabātā ziņa nepareizi nolasītos vai nebūtu pieejama. Tāpēc, LSB steganogrāfijai ir ieteicams izmantot "lossless compression" formāta bildes, lai saglabātos RGB vērtību precizitāte.[8]

1.7 "Lossy compression" metodes bilžu formāts.

Šādu metodi pielieto JPEG bilžu formāts un tā paveidi, lai augstas kvalitātes bildes saglabātos, aizņemot pēc iespējas mazāk vietas un resursus. Šī metode izmanto "Chroma subsampling" algoritmu, kas dotajai bildei salīdzina pikseļus savā starpā un saglabā izlecošās krāsu vērtības attēlā, sapludinot tumšos krāsu pikseļus kopā, lai panāktu pēc iespējas mazāk atšķirīgas pikseļu vērtības. Rezultātā samazinās datu daudzums un faila lielums. Šajā metodē var arī izvēlēties attēla saspiešanu lai tas aizņemtu pēc iespējas mazāk atmiņas, vai pretēji – saglabājot attēlu ar pēc iespējas labāku kvalitāti, bet palielinot faila lielumu.[9]

Šo iemeslu dēļ šāds faila tips nav ieteicams, lai veiktu LSB steganogrāfiju, jo iešifrētais teksts kompresijas laikā varētu pazust vai manīties, failam veicot savu algoritmu uzreiz pēc faila saglabāšanas.

1.8 "Lossless compression" metodes bilžu formāts

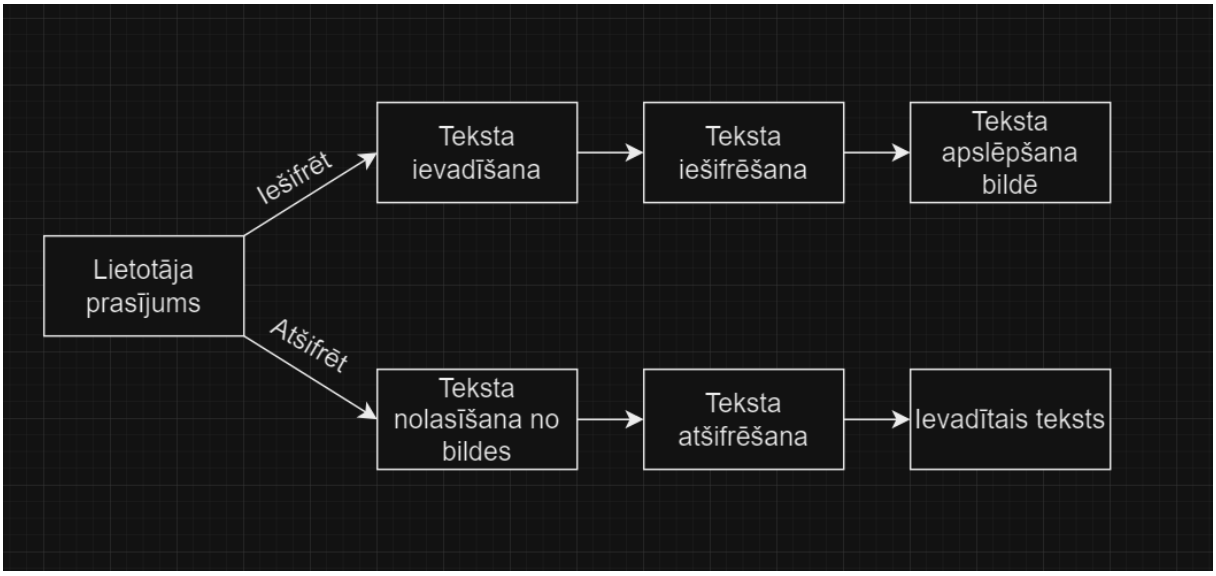
Šāda metode saglabā bildes kvalitāti, kāda tā ir dota, bet izmantojot algoritmus, samazina tā lielumu, tomēr mazākā apjomā, salīdzinot ar "Lossy compression" metodi. Visbiežāk failu formāti, kuros sastopama šāda metode ir PNG vai BMP faili. "Lossless compression" algoritms strādā, meklējot failā bieži atkārtotus datus, saglabājot tos kā referenci un izmantojot šo referenci atkārtotas datu pārrakstīšanas vietā. Šī metode pēc iespējas vairāk garantē, ka bildes kvalitāte saglabāsies, un ka pikseļu vērtības netiks mainītas, lai aizņemtu mazāk atmiņas.[10]

Šī metode ir noderīgāka LSB steganogrāfijas metodei, kur katra pikseļa krāsas vērtības ir saglabātas tā, lai tās atbilstu tam, kā tās tika mainītas iešifrēšanas rezultātā.

2. Programmatūras izveide

Šīs programmatūras izstrādei nepieciešama izpratne par kriptogrāfijas un steganogrāfijas principiem, lai varētu izvēlēties sev vairāk atbilstošu metožu pielietojumu iecerētajam rezultātam, tā pielietojumam. Programmatūru var izveidot dažādās valodās un metodēs. Ir nepieciešams zināt konkrētu mērķi programmatūrai, lai izvēlētos atbilstošāko valodu.

Programmatūra sadalīta 2 daļās, kur viena veic iešifrēšanu un atšifrēšanu, un otra daļa veic apslēpšanu bildē un bildes nolasīšanu. Ir svarīgi iegaumēt secību, kurā veiktas šīs darbības. No sākuma veikta teksta iešifrēšana, tikai pēc tam tiek veikta teksta apslēpšana bildē. Kad pieprasīta teksta atšifrēšana, tiek nolasīts teksts no bildes un tikai tad tas tiek atšifrēts(skat. 4. attēlu).



(4. attēls. Vienkāršota programmatūras darbības vizualizācija, apvienojot kriptogrāfijas un steganogrāfijas metodes)

Veicot programmatūras izstrādi, jāņem vērā tās sarežģītība, jo dažādos gadījumos var atšķirties nepieciešamība pēc sarežģītības. Iešifrēšanas procesa sarežģītība ir jāņem vērā programmatūras autoram, jo laiks iešifrēšanas un atšifrēšanas procesam palielinās ar katru pievienoto sarežģījumu. Ir svarīgi sadalīt metodes tā, lai tās atbilstu to pielietojumam, jo ir teksti, kuriem nav nepieciešama sarežģīta algoritma izveide, jo to noplūde neizraisa liela mēroga riskus.

2.1 Programmatūras izveides process

Darbam izvēlēta “Python” programmēšanas valoda. Valoda izvēlēta, pamatojoties uz tās plašo pielietojumu un autora priekšzināšanām. Programmatūras izstrādei nav nepieciešamas funkcijas, kas nav pieejamas “Python” valodā. Lai utilizētu izstrādāto programmatūru un ievietotu tajā tekstu, darbam izvēlēts “Visual Studio Code” programmatūras “code compilers” rīks, kurš paredzēts koda rakstīšanai un palaišanai vairākās programmēšanas valodās. Šis “Code compilers” atbalsta dažādas programmēšanas valodas, iekļaujot “Python”.

Programmatūras izstrādei izvēlētas attiecīgās kriptogrāfijas un steganogrāfijas metodes. Šim nolūkam tika izmantotas dažādas bibliotēkas metožu realizācijai. Darbā pielietota simetriskā kriptogrāfija, jo šai metodei nav nepieciešams daudz resursu, salīdzinot ar asimetrisko kriptogrāfiju. Asimetriskajai kriptogrāfijai vajadzētu izveidot savienojumu starp ierīcēm, kuras komunicē caur serveri vai citiem veidiem, kas ietver autoram nepieejamu līdzekļu izmantošanu. Šī metode bija praktiskāka, jo to var pielietot jebkurā sociālajā tīklā, pārsūtot savu apslēpto tekstu, bez vajadzības atstāt ierīci darbojoties nepārtraukti. Šī metode tika pielietota "Python" izstrādātas bibliotēkas kriptogrāfijai "cryptography". No visām piedāvātajām funkcijām izvēlēta tieši "Fernet" metode. Teksta apslēpšanai bildēs pielietota LSB steganogrāfijas metode, jo šai metodei bija vispieejamākie resursi dokumentācijai un paskaidrojumiem tās pielietošanai. Tika izvēlēta bibliotēka "pillow", kas veidota "Python" valodai, un pieļauj LSB steganogrāfijas metodes izstrādi.

Sākotnēji izveidota iespēja lietotājam izvēlēties darbību, kuru programma veiks. Tā var būt iešifrēšana, atšifrēšana vai jaunas paroles izveide. Pēc lietotāja izvēles, uzreiz izpildītas prasītās darbības(skat. 5.attēlu).

```
choice = input("IF YOU HAVENT GENERATED A PASSWORD YET, THEN USE CHOICE 3.\nEncrypt - 1\nDecrypt - 2\nGenerate New Password - 3\n")
if choice == "1":
    message = input("Enter Your message: ")
    password_path = input("Enter the password file path: ")
    with open(password_path, "rb") as file:
        password = file.read()
        file.close()
    path = input("Enter the path of the image: ")
    text = encrypt(message, password)
    print(text)
    hide_text_in_image(text, path)
elif choice == "2":
    path = input("Enter the password file path: ")
    with open(path, "rb") as file:
        password = file.read()
        file.close()
    msg = extract_text_from_image()
    text = decrypt(msg, password)
    print("Extracted Text:", text)
elif choice == "3":
    password = generate_key()
    file_path = "password.dat"
    with open(file_path, "wb") as file:
        file.write(password)
        file.close()
```

(5. attēls. Kods, kurš, atkarībā no lietotāja izvēles, ievadot attiecīgo skaitli veiks nepieciešamo funkciju, kur 1 atbilst "Iešifrēt", 2 "Atšifrēt" un 3 "Izveidot paroli".)

Izveidota funkcija paroles izveidei atbilstoši iešifrēšanas metodei, kura tiks pielietota iešifrēšanas un atšifrēšanas laikā. Šī funkcija veido paroli, kas, izmantojot konkrētos un modernos algoritmus, atbilst drošības standartiem un algoritma darbībām.

```
def generate_key():
    return Fernet.generate_key()
```

(6. attēls. Kods no programmatūras paroles izveidei.)

Tālāk izveidotas teksta iešifrēšanas un atšifrēšanas funkcijas (skat. 7. attēlu), kurās pielietota "cryptography" bibliotēka, kura aprastīta iepriekš. Šajās funkcijās uzrakstīts algoritms, kurš atsaucas uz "cryptography" bibliotēku, lai pielietotu tajās izmantotās metodes teksta atšifrēšanai un iešifrēšanai.

```python
def encrypt(message, key):
    cipher = Fernet(key)
    encrypted_message = cipher.encrypt(message.encode())
    return encrypted_message

def decrypt(encrypted_message, key):
    cipher = Fernet(key)
    decrypted_message = cipher.decrypt(encrypted_message).decode()
    return decrypted_message
```

(7. attēls. Iešifrēšanas un atšifrēšanas funkcijas programmatūras kodā.)

Tālāk izveidotas funkcijas, kas pārveido datus no bitiem uz bināro kodu un pretēji. Kriptogrāfijas rezultātā šifrētais teksts pārveidots par bitiem, bet, LSB steganogrāfijas pielietojumā, teksts pārtop binārajā kodā. Šīs funkcijas izveidotas, lai neveidotos konflikti starp funkcijām (skat. 8 attēlu).

```python
def bytes_to_binary(text):
    binary = ''.join(format(byte, '08b') for byte in text)
    return binary

def binary_to_bytes(binary_string):
    byte_array = bytearray(int(binary_string[i:i+8], 2) for i in range(0, len(binary_string), 8))
    return bytes(byte_array)
```

(8. attēls. Kodā izmantotās funkcijas datu pārtulkošanai no bitiem uz bināro kodu.)

Pēdējās funkcijas ir steganogrāfijas funkcijas (skat. 9. un 10. attēlus), kuras pielieto iepriekš minēto LSB steganogrāfijas metodi. Tās lietotas gan teksta šifrēšanai, gan šifrētā teksta nolasīšanai no attēla.

```python
def hide_text_in_image(text, path):
    image_path = path
    text_to_hide = text

    binary_text = bytes_to_binary(text_to_hide)

    img = Image.open(image_path)
    width, height = img.size

    if len(binary_text) > (width * height * 3):
        raise ValueError("Text is too long to be hidden in this image.")

    binary_text += '1111111111111110'

    data_index = 0
    for y in range(height):
        for x in range(width):
            pixel = list(img.getpixel((x, y)))

            for color_channel in range(3):
                if data_index < len(binary_text):
                    pixel[color_channel] = pixel[color_channel] & 254 | int(binary_text[data_index])
                    data_index += 1

            img.putpixel((x, y), tuple(pixel))

    output_path = input("Enter the path for the output image (include .png extension): ")
    img.save(output_path)
    print("Text hidden in the image and saved as", output_path)
```

(9. attēls. Funkcija teksta apslēpšanai attēlā.)

```python
def extract_text_from_image():
    image_path = input("Enter the path of the image from which to extract text: ")
    img = Image.open(image_path)
    binary_text = ''
    delimiter = '1111111111111110'

    for y in range(img.height):
        for x in range(img.width):
            if binary_text[-len(delimiter):] == delimiter:
                break
            pixel = img.getpixel((x, y))
            for color_channel in pixel:
                binary_text += str(color_channel & 1)
                if binary_text[-len(delimiter):] == delimiter:
                    break

    delimiter_index = binary_text.find(delimiter)
    if delimiter_index != -1:
        binary_text = binary_text[:delimiter_index]

    text = binary_to_bytes(binary_text)
    print(f"The extracted text is: {text}")
    return text
```

(10. attēls. Funkcija teksta nolasīšanai no attēla.)

2.2 Programmatūras rezultāta apraksts

Izstrādes rezultātā, programmatūra sastāv no 7 funkcijām, kur divas no tām iekļauj teksta iešifrēšanu un atšifrēšanu, izmantojot uzģenerētu paroli. Divas funkcijas pārveido bitus uz bināro kodu un pretēji, divas apslēpj attēlā tekstu un to nolasa, un viena uzģenerē un nolasa paroli pēc pieprasījuma.

Programmatūra atbilstoši izveides plānam, kur lietotājs uzģenerē sev lietojamu paroli teksta šifrēšanai, un izmanto šo paroli, lai nosūtītu iešifrētus tekstus bildēs. Vienīgais, kas ir nepieciešams, lai tekstus atšifrētu, ir dotā parole. Lietotājam ir iespēja mainīt paroli jebkurā brīdī. Tekstus nebūs iespējams atšifrēt ar citu paroli.

Lai lietotājs varētu pielietot programmu, viņam ir nepieciešams lejupielādēt vienu bildi, kurā viņs vēlētos apslēpt tekstu, ar nosacījumu, ka attēlā nav pikseļi bez vērtībām.

Lai apslēptu bildē šifrētu tekstu, lietotājs ievada attēla atrašanās vietu ar tās nosaukumu un faila formātu. Tālāk, lietotājs ievada faila nosaukumu, kurā atrodas uzģenerētā parole. Nobeidzot, lietotājs ievada tekstu, kuru viņš vēlētos apslēpt.

Darbību beigās, lietotājs gūs iespēju nosaukt izveidoto failu ar apslēpto tekstu, ar nosacījumu, ka tiek izmantots “PNG” formāts. Formāts nepieciešams, lai netiktu zaudētas pikseļu vērtības pārsūtīšanas laikā. Rezultātā izveidots fails ar ievadīto nosaukumu, kurā iekļauta iešifrētā bilde.

Lai attēlu atšifrētu, lietotājam ir nepieciešama piekļuve bildei, kuru viņš vēlas atšifrēt, paroles fails, kurā tiek glabāta izmantotā parole iešifrēšanā. Palaižot programmu, lietotājam tiek nosūtīts pieprasījums norādīt atšifrējamās bildes un paroles faila atrašanās vietu un nosaukumu, kuras rezultātā terminālī tiks izprintēts atšifrētais teksts.

Secinājumi

1. Ir sastopamas 3 kriptogrāfijas metodes, kuras var pielietot, atkarībā no programmatūras nepieciešamības un izstrādātāja mērķiem.
2. Ir sastopami dažādi steganogrāfijas pielietojumi dažādās vidēs, kas ļauj pielietot gandrīz jebko, lai apslēptu datus ikdienas lietās.
3. Ir iespējams izstrādāt ļoti sarežģītu algoritmu, kas būtu gandrīz vai neuzlaužams, bet, lai to pielietotu efektīvi, ir nepiecišami lielāki resursi un materiāli, kā arī ilgāks laiks, lai to iešifrētu un atšifrētu.
4. Attēla failos ir iespējams apslēpt informāciju dažādos veidos, kas var būt modificējot faila nosaukumu, failā ierakstītos datus un bilžu kvalitāti.
5. Lai pielietotu LSB steganogrāfijas metodi ir nepieciešams ņemt vēra bildes dimensiju izmērus, jo tas nosaka tās datu ietilpību.

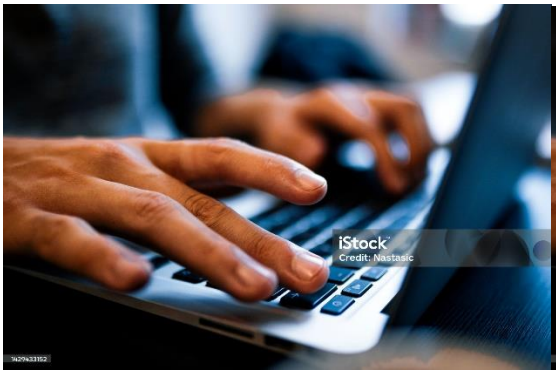
Uzstādītos darba mērķus izdevās sasniegt, jo programmatūras izstrādes rezultātā autors iepazinās ar kriptogrāfijas un steganogrāfijas pielietojumu programmas izveidei. Tika izstrādāta programmatūra, kas apvieno iepriekš pieminētās metodes, lai atšifrētu un apslēptu datus attēlos, veicot nemanāmas izmaiņas. Nepieciešamības gadījumā, šo programmatūru lietotāji var pēc izvēles pielāgot un arī viegli utilizēt.

Literatūras saraksts

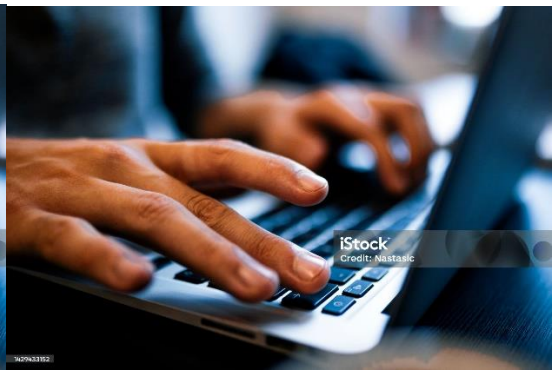
1. - Geeks for Geeks, achivchauhan, Introduction to Crypto-terminologies. Pieejams: https://www.geeksforgeeks.org/introduction-to-crypto-terminologies/. [Skatīts: 12.12.2023]
2. – IBM, Symmetric crptography. Pieejams: https://www.ibm.com/docs/en/ztpf/2020?topic=concepts-symmetric-cryptography [Skatīts: 12.12.2023]
3. – TechTarget, (Kate Brush, Linda Rosencrance, Michael Cobb), Asymmetric cryptography (public key cryptography). Pieejams: https://www.techtarget.com/searchsecurity/definition/asymmetric-cryptography [Skatīts: 19.12.2023]
4. – SentinelOne, What is hashing?. Pieejams: https://www.sentinelone.com/cybersecurity-101/hashing/ [Skatīts: 26.12.2023]
5. - TechTarget, Margie Semilof , steganography. Pieejams: https://www.techtarget.com/searchsecurity/definition/steganography [Skatīts: 20.12.2023]
6. – Kaspersky, What is steganography? Definition and explanation. Pieejams: https://www.kaspersky.com/resource-center/definitions/what-is-steganography [Skatīts: 27.12.2023]
7. – Research Gate, Mehdi Hussain, A Survey of Image Steganography Techniques. Pieejams: https://www.researchgate.net/publication/271863505_A_Survey_of_Image_Steganography_Techniques [Skatīts: 27.12.2023]
8. - Int. J. Advanced Networking and Applications,(V. Lokeswara Reddy, Dr. A. Subramanyam, Dr.P. Chenna Reddy), Implementation of LSB Steganography and its Evaluation for Various File Formats. Pieejams: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3dceb6307cee042b687b7f377ec1d5de91ce20b0 [Skatīts: 29.12.2023]
9. – Khan Academy, Lossy compression. Pieejams: https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:digital-information/xcae6f4a7ff015e7d:data-compression/a/lossy-compression [Skatīts: 29.12.2023]
10. – Adobe, What is lossless Compression? Everything we need to know. Pieejams: https://www.adobe.com/uk/creativecloud/photography/discover/lossless-compression.html [Skatīts: 29.12.2023]

Pielikums

1. Pielikums.
   Programmas kods, kurš bija izveidots praktiskajā daļā - https://github.com/NotErnests/ZPD_Steganography_and_cryptography_program
2. Pielikums.
   Programmatūras rezultātā izveidotā bilde, kurā ir apslēpts teksts salīdzinājumā ar sākumā ievadīto.



(1.attēls Programmatūrai pasniegtā bilde)



(2.attēls Rezultātā izvadītā bilde)