

# Kriptografija

## Duomenų tyrybos, programavimo ir saugaus elgesio pradmenys

Parengta remiantis Lietuvos  
informatikos mokytojų asociacijos  
rekomendacijomis

2025 m.

# Kriptografinių sistemų apibūdinimas

- ❑ **Kriptografija** – iš graikų kalbos **kripto** (*paslėptas*) ir **grafos** (*rašymas*).
- ❑ Kriptografija reiškia paslėptą rašymą, kurio **tikslas** aiškus – taip norima **apsaugoti informaciją** ir neleisti trečiosioms šalims prie jos prieiti.
- ❑ **Kriptografinės sistemos** – tai matematikos ir informatikos disciplinos, kurioje nagrinėjami duomenų užšifravimo ir šifro atkūrimo (iššifravimo) metodai.
- ❑ **Jų tikslas** – užtikrinti informacijos saugumą nuo nepageidaujamų asmenų (nuo asmenų, kurie neturi teisės matyti ar modifikuoti šiuos duomenis).



# Pagrindinės šiuolaikinių kriptografinių sistemų kategorijos

- ❑ **Simetrinė kriptografija** – tai kriptografijos tipas, kai **tie patys šifravimo raktai** naudojami tiek užšifravimui, tiek iššifravimui. Taip šifruojant, pranešimo siuntėjas ir gavėjas turi turėti tą patį raktą, kad užšifruotas pranešimas būtų perskaitytas.
- ❑ **Asimetrinėje kriptografijoje** – pranešimo užšifravimui ir iššifravimui šiame **naudojami skirtingi raktai**. Paprastai yra **viešasis raktas**, kuris yra prieinamas visiems ir naudojamas užšifravimui, bei **privatūs raktai**, kurie naudojami tik iššifravimui ir yra laikomi paslapyje.
- ❑ **Hibridinė kriptografija** kombinuoja simetrinės ir asimetrinės kriptografijos privalumus. Paprastai naudojama asimetrinė kriptografija saugiam simetrinio rakto perdavimui, o po to su šiuo simetriniu raktu šifruojami duomenys.

# Kurį pasirinkti?

- ❑ **Simetrinis šifravimas** yra greitas ir paprastas, bet kyla problemų, kai reikia saugiai pasidalyti ir saugoti raktus.
- ❑ **Asimetrinis šifravimas** šias problemas sprendžia, bet yra lėtesnis ir reikalauja daugiau kompiuterio resursų.

\*\*\*\*\*

**Kriptografinės sistemos naudojamos** daugelyje modernių technologijų, tokių kaip elektroninis paštas, interneto svetainės, elektroninių mokėjimų sistemos, virtualūs privatūs tinklai (VPT), kriptovaliutos, karyba ir kt.

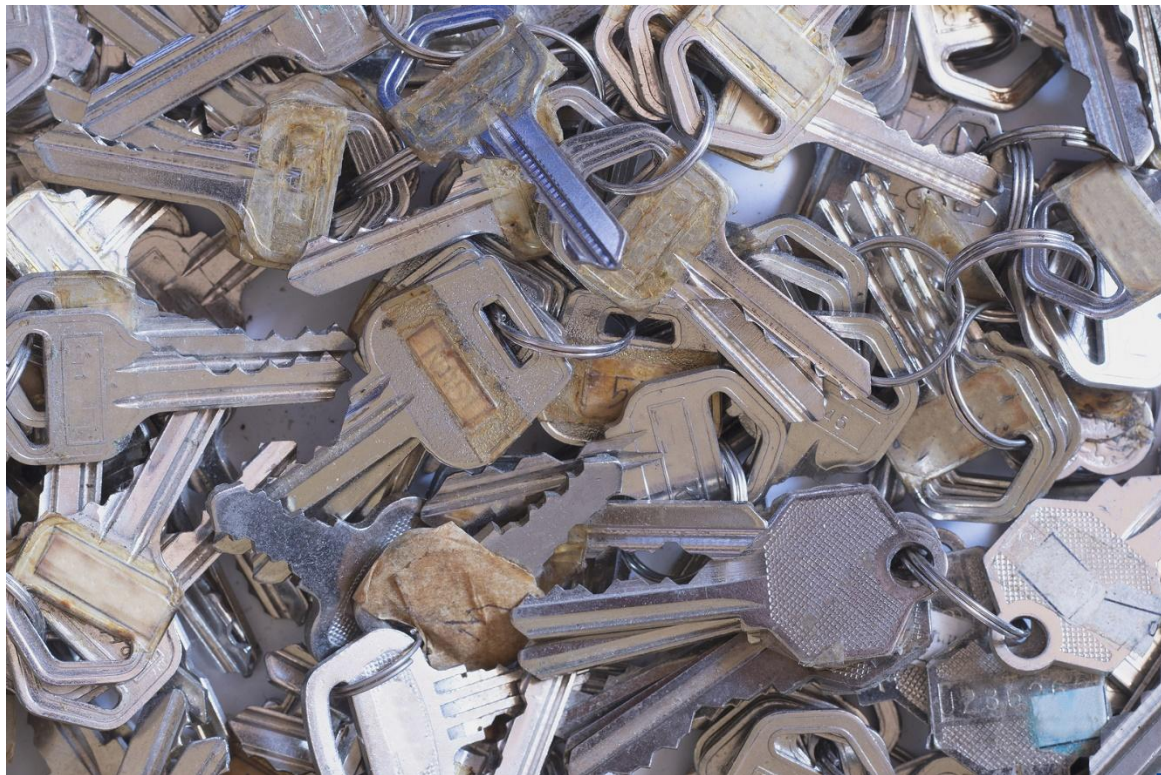
# Simetrinė kriptografija gali būti skirstoma

- ❑ **Šifravimas nenaudojant kompiuterių** – dažniausiai yra istoriniai šifravimo metodai, kurie buvo naudojami prieš kompiuterių erą, pavyzdžiui, *Cezario šifras*, *Skytale*, *Knyginis šifras*, *Perstatų šifras* ir kt.
- ❑ **Šifravimas naudojant kompiuterius**, pavyzdžiui, *Advanced Encryption Standard* (AES – *Išplėstinis šifravimo standartas*), kuris yra vienas iš populiariausių ir plačiausiai naudojamų simetrinio šifravimo algoritmų.

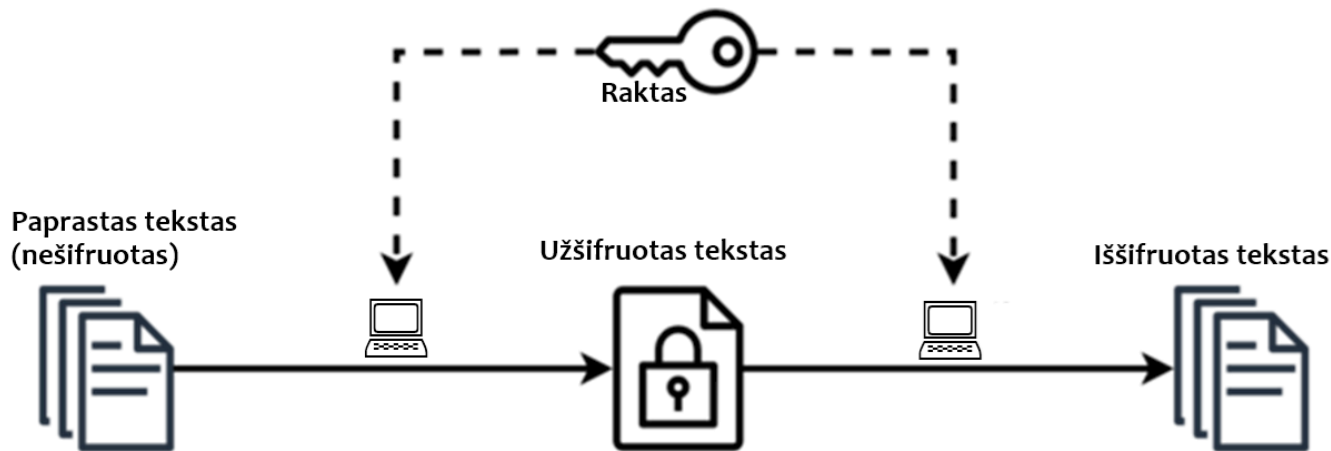
# Simetrinis šifravimas (kriptografija)

## Problema

Kuo daugiau žmonių turės jūsų suteiktą raktą, tuo didesnė bus saugos rizika.




# Simetrinio šifravimo naudojant kompiuterius schema



## Užduotis

Simetrinio šifravimo atveju sauga ypatingai priklauso nuo šifravimo rakto (metodo), šio rakto saugojimo ir jo saugaus perdavimo komunikacijos dalyviui. Pakomentuokite simetrinio šifravimo galimas saugos spragas ir veiksmus, didinančius šifravimo saugą.





Simetrinė kriptografija turi įvairių taikymų sričių, kuriose ji gali būti vertinama dėl savo greitaveikos ir efektyvumo.

# Simetrinė kriptografija – taikymas (1)

## Duomenų saugojimas

- ❑ Šifruoti standieji diskai (naudojama siekiant apsaugoti duomenis diskuose nuo neautorizuotos prieigos).
- ❑ Duomenų saugojimo debesijos paslaugos (kai kurie tiekėjai leidžia klientams šifruoti savo failus prieš juos įkeliant į debesį).

## Komunikacija:

- ❑ Šifruotos žinutės ir pokalbiai (kai kurios žinučių siuntimo programėlės naudoja simetrinę kriptografiją šifruoti pokalbiams ir nuotraukoms).
- ❑ Saugus VoIP (Voice over IP) (simetrinė kriptografija gali būti naudojama balso ir vaizdo pokalbiams šifruoti).

# Simetrinė kriptografija – taikymas (2)

## Tinklų saugumas

- ❑ Wi-Fi saugumas (WPA ir WPA2 – bevielio tinklo saugumo protokolai naudoja simetrinę kriptografiją (pavyzdžiui, AES) saugumui užtikrinti).
- ❑ VPN (VPT – *virtualūs privatūs tinklai*) paslaugos (simetrinė kriptografija yra naudojama greitai ir saugiai perduoti duomenis).

## Finansų sektorius

- ❑ Elektroniniai mokėjimai (simetrinė kriptografija yra naudojama piniginių transakcijų duomenims saugoti).
- ❑ ATM transakcijos (kai kurios bankomato operacijos gali naudoti simetrinę kriptografiją saugoti piniginei informacijai).

# Simetrinė kriptografija – taikymas (3)

## **Pramoninės ir kontrolinės sistemos**

- ❑ SCADA ir ICS (naudojama simetrinę kriptografiją saugoti komunikacijai tarp valdymo centų ir nuotolinių įrenginių).

## **Elektroninė prekyba**

- ❑ Saugus duomenų perdavimas (simetrinė kriptografija gali būti naudojama užtikrinti konfidencialumą perduodant klientų informaciją ir mokėjimo duomenis).

## **Šifruotas IPTV ir kiti šifruoti transliacijos protokolai**

- ❑ Naudojama apsaugoti nuo neautorizuotos prieigos prie turinio.

# Simetrinė kriptografija – taikymas (4)

## Automobilių pramonė

- ❑ Saugus nuotolinis valdymas (simetrinė kriptografija gali būti naudojama saugiai komunikacijai tarp nuotolinio rakto ir automobilio užtikrinti).

## Medicinos įrenginiai

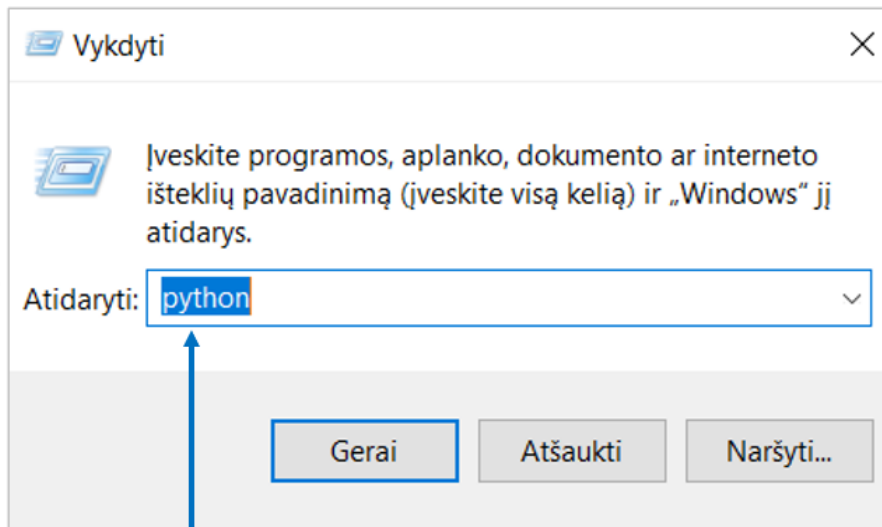
- ❑ Paciento duomenų apsauga (simetrinė kriptografija gali būti naudojama saugoti jautrius paciento duomenis medicinos įrenginiuose).

**Ir kitos taikymo sritys.**

Panagrinėkime pavyzdžius, kaip gali būti panaudota programavimo kalba *Python* kuriant kompiuterio programas, naudojančias simetrinio *Advanced Encryption Standard (AES)* šifravimo biblioteką tekstui užšifruoti ir iššifruoti.

# Pasiruošimas (1)

Klaviatūroje  
paspauskite:



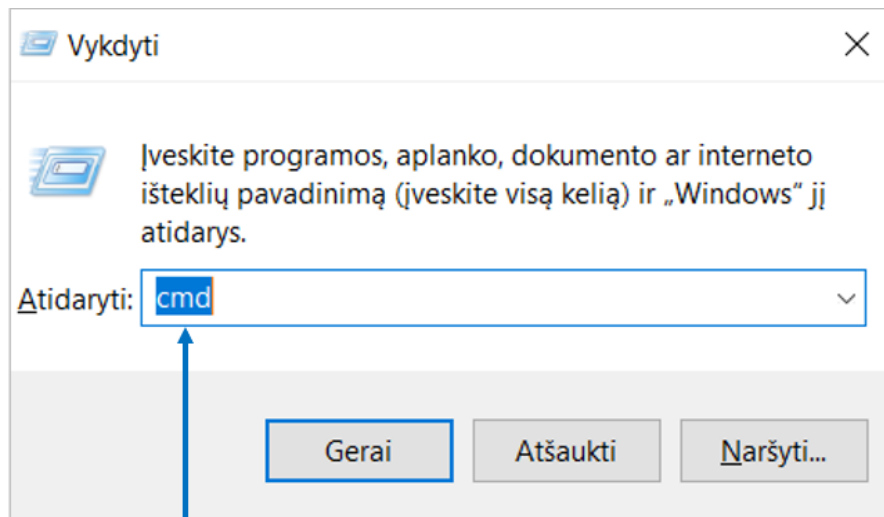
Atsivėrusiame langelyje parašykite  
**python** ir paspauskite klavišą **Enter ↵**



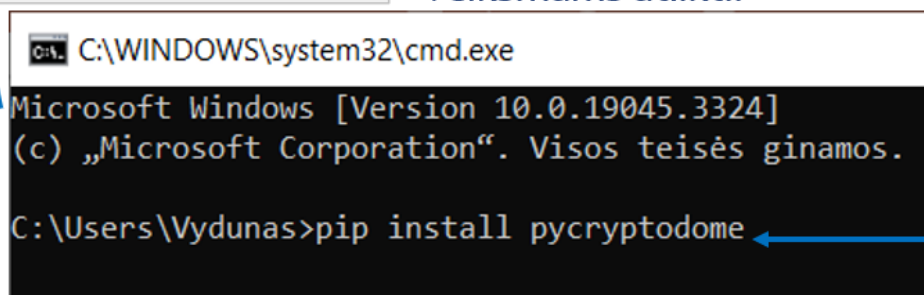
Instaliuojama  
Python programa

# Pasiruošimas (2)

Klaviatūroje paspauskite:



Atsivėrusiame langelyje parašykite **cmd** ir paspauskite klavišą **Enter**



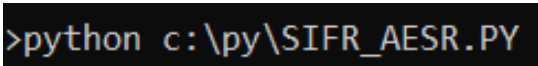
Pereinama į Windows komandų eilutę. Joje parašykite (nukopijuokite) ***pip install pycryptodome*** ir paspauskite **Enter**

Taip įdiegiama Python biblioteka **PyCryptodome**, reikalinga šifravimo veiksams atlikti.

Kopijuoti: **pip install pycryptodome**



# Išbandykite simetrinio užšifravimo įrankį (1)

- ❑ Nusikopijuokite čia pateiktą programos kodą į „Užrašinę“ ir įrašykite vardu **SIFR\_AESR.PY** į **C:** disko aplanką **py**.
- ❑ Įvykdykite programą, komandų eilutėje nurodydami\* **python c:\py\SIFR\_AESR.PY**  

- ❑ Panagrinėkite programos darbo rezultatą – informaciją, pateiktą įvykdžius programą (žr. kitą skaidrę).

\***Pastaba.** Jei įrašysite programą kitų vardu ir (arba) kitame aplanke, komandų eilutėje rašykite savo informaciją – nurodykite savo aplanką ir savo failo vardą.

```
# PROGRAMOS SIFR_AESR.PY KODAS
from Crypto.Cipher import AES
```

```
def main():
    # Naudokime pastovų 32 baitų ilgio raktą
    key = b"ThisIsASecretKey_32bytesLongAbCd"

    # Inicijuoti AES šifro objektą su EAX režimu
    cipher = AES.new(key, AES.MODE_EAX)

    # Teksto užšifravimas
    plaintext = b"Slapta: Kompiuteris yra tavo ir mano geriausias draugas :-)."
    ciphertext, tag = cipher.encrypt_and_digest(plaintext)

    # Atspausdiname raktą, užšifruotą tekstą, tag ir nonce reikšmes
    print(f"Raktas: {key.decode('utf-8')}")
    print(f"Užšifruotas tekstas: {ciphertext.hex()}")
    print(f"Tag: {tag.hex()}")
    print(f"Nonce: {cipher.nonce.hex()}") # Atspausdiname nonce reikšmę

    # Teksto iššifravimas
    cipher_dec = AES.new(key, AES.MODE_EAX, nonce=cipher.nonce)
    decrypted = cipher_dec.decrypt(ciphertext)

    # Patikriname, ar iššifruotas tekstas sutampa su pradiniais duomenimis
    try:
        cipher_dec.verify(tag)
        print(f"Pradinis: {decrypted.decode('utf-8')}")
    except ValueError:
        print("Iššifruotasis tekstas yra neteisingas!")

if __name__ == '__main__':
    main()
```

# Išbandykite simetrinio užšifravimo įrankį (2)

- ❑ Įvykdytos programos **SIFR\_AESR.PY** darbo rezultatas:

```
Raktas: ThisIsASecretKey_32bytesLongAbCd
Užšifruotas tekstas: e7883d8abd0243e2636be5c52899b1d3c749ba080807a22c959394750e24d72d35547cd15e570627517c700e1148f590208f79f7f7afee7fbe7076073
Tag: 733d3cfd1421fdc76095bc8f65175d73
Nonce: b05e6383dee9e5e6406f112795d7276c
Pradinis: Slapta: Kompiuteris yra tavo ir mano geriausias draugas :-).
```

- ❑ Rezultato teksto dalys: **Raktas**, **Užšifruotas tekstas**, **Tag**, **Nonce**, **Pradinis**.
- ❑ Apie **Nonce** ir **Tag**:
  - **Nonce** garantuoja, kad tam pačiam pranešimui, net ir naudojant tą patį raktą, bus sugeneruotas skirtingas šifro tekstas kiekvieno šifravimo metu.
  - **Tag** yra simbolių seka, kuri garantuoja pranešimo autentiškumą ir integralumą. Ji leidžia gavėjui patikrinti ir įsitikinti, ar šifruotasis pranešimas arba duomenys nebuvo pažeisti ar pakeisti.
  - **Išvada:** **Nonce** užtikrina, kad kiekvienas šifro tekstas yra unikalus, o **Tag** patvirtina, kad šifruotasis pranešimas yra autentiškas ir nepažeistas.

# Išbandykite simetrinio užšifravimo įrankį (3)

- ❑ Užšifruokite kitą tekstą – įrašykite norimą užšifruoti tekstą\* atitinkamoje pateiktos programos vietoje ir įvykdysite programą.
- ❑ Įvykdytos programos su kitu užšifruojamu tekstu darbo rezultatas:

```
Raktas: ThisIsASecretKey_32bytesLongAbCd
Užšifruotas tekstas: dbb6fd975fafc6282d8a27dd9841cc26bf67929926e7648fa
0b50fdc88bff2641ab5e68b7c447f310d527b6ff2df1f5c4498c4be4ede13
Tag: 265df1dfa03b68cd368605e2660c763d
Nonce: 78db0191dcadb80bb318579ad3a535d3
Pradinis: Kas nedarba, mielas vaike, tam ir duonos duot nereikia.
```

\***Pastaba.** Šioje užšifravimo kompiuterio programoje naudokite tik ASCII kodo raides (raides be diakritinių ženklų).

```
# PROGRAMOS SIFR_AESR.PY KODAS
from Crypto.Cipher import AES
```

```
def main():
    # Naudokime pastovų 32 baitų ilgio raktą
    key = b"ThisIsASecretKey_32bytesLongAbCd"
```

```
# Inicijuoti AES šifro objektą su EAX režimu
cipher = AES.new(key, AES.MODE_EAX)
```

```
# Teksto užšifravimas
plaintext = b"Kas nedarba, mielas vaike, tam ir duonos duot nereikia."
ciphertext, tag = cipher.encrypt_and_digest(plaintext)
```

```
# Atspausdiname raktą, užšifruotą tekstą, tag ir nonce reikšmes
print(f"Raktas: {key.decode('utf-8')}")
print(f"Užšifruotas tekstas: {ciphertext.hex()}")
print(f"Tag: {tag.hex()}")
print(f"Nonce: {cipher.nonce.hex()}") # Atspausdiname nonce reikšmę
```

<...>

Čia pateiktas programos teksto fragmentas

Nukopijuokite programos darbo rezultato informaciją į, pavyzdžiui, programą „Užrašinė“ ir įrašykite failą– šią informaciją panaudosite iššifruodami užšifruotą pranešimą (žr. kitą skaidrę).

# Išbandykite simetrinio iššifravimo įrankį (1)

- ❑ Nusikopijuokite čia pateiktą programos kodą į užrašinę.
- ❑ Į programos teksto atitinkamas vietas įkelkite anksčiau išsaugotą užšifravimo informaciją\*:
  - Raktas → **key**
  - Užšifruotas tekstas → **ciphertext\_hex**
  - Tag → **tag\_hex**
  - Nonce → **nonce\_hex**
- ❑ Įrašykite programą vardu **SIFR\_AESS.PY** į **C:** disko aplanką **py**.

\*Pavyzdyje panaudota informacija, gauta užšifravus tekstą  
*Kas nedirba, mielas vaike, tam ir duonos duot nereikia.*

```
# PROGRAMOS SIFR_AESS.PY KODAS
from Crypto.Cipher import AES
from binascii import unhexlify

def decrypt_aes_eax(ciphertext_hex, key, tag_hex, nonce_hex):
    # Konvertuojame šiuos duomenis iš šešiolyktainės į baitų sekas
    ciphertext = unhexlify(ciphertext_hex)
    tag = unhexlify(tag_hex)
    nonce = unhexlify(nonce_hex)

    # Sukuriamas šifravimo objektas su pateiktu raktu ir nonce
    cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)

    # Bandoma iššifruoti tekstą ir patikrinti MAC su pateiktu tag'u
    try:
        decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
        return decrypted_data.decode('utf-8')
    except ValueError:
        return "Užšifruotas tekstas buvo pažeistas. Iššifravimas nutrauktas."

def main():
    # Pateikti duomenys
    key = b"ThisIsASecretKey_32bytesLongAbCd"
    ciphertext_hex =
    "dbb6fd975fafc6282d8a27dd9841cc26bf67929926e7648fa0b5ofdc88bff26
    41ab5e68b7c447f310d527b6ff2df1f5c4498c4be4ede13"
    tag_hex = "265df1dfa03b68cd368605e2660c763d"
    nonce_hex = "78db0191dcadb80bb318579ad3a535d3"

    decrypted_message = decrypt_aes_eax(ciphertext_hex, key, tag_hex,
    nonce_hex)
    print(f'Iššifruotas tekstas: {decrypted_message}')

if __name__ == '__main__':
    main()
```

# Išbandykite simetrinio iššifravimo įrankį (2)

- ❑ Įvykdykite programą, komandų eilutėje nurodę\*:

**python c:\py\SIFR\_AESS.PY**

```
>python c:\py\SIFR_AESR.PY
```

- ❑ Programos darbo rezultatas – iššifruotas pranešimas:

Iššifruotas tekstas: Kas nedirba, mielas vaike, tam ir duonos duot nereikia.

**\*Pastaba.** Jei įrašysite programą kitų vardu ir (arba) kitame aplanke, komandų eilutėje rašykite savo informaciją – nurodykite savo aplanką ir savo failo vardą.

```
# PROGRAMOS SIFR_AESS.PY KODAS
from Crypto.Cipher import AES
from binascii import unhexlify

def decrypt_aes_eax(ciphertext_hex, key, tag_hex, nonce_hex):
    # Konvertuojame šiuos duomenis iš šešiolyktainės į baitų sekas
    ciphertext = unhexlify(ciphertext_hex)
    tag = unhexlify(tag_hex)
    nonce = unhexlify(nonce_hex)

    # Sukuriamas šifravimo objektas su pateiktu raktu ir nonce
    cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)

    # Bandoma iššifruoti tekstą ir patikrinti MAC su pateiktu tag'u
    try:
        decrypted_data = cipher.decrypt_and_verify(ciphertext, tag)
        return decrypted_data.decode('utf-8')
    except ValueError:
        return "Užšifruotas tekstas buvo pažeistas. Iššifravimas nutrauktas."

def main():
    # Pateikti duomenys
    key = b"ThisIsASecretKey_32bytesLongAbCd"
    ciphertext_hex =
    "dbb6fd975fafc6282d8a27dd9841cc26bf67929926e7648fa0b50fdc88bff26
    41ab5e68b7c447f310d527b6ff2df1f5c4498c4be4ede13"
    tag_hex = "265df1dfa03b68cd368605e2660c763d"
    nonce_hex = "78db0191dcadb80bb318579ad3a535d3"

    decrypted_message = decrypt_aes_eax(ciphertext_hex, key, tag_hex,
    nonce_hex)
    print(f"Iššifruotas tekstas: {decrypted_message}")

if __name__ == '__main__':
    main()
```

# Išbandykite simetrinio iššifravimo įrankį (3)

- ❑ Išstirkite, koks bus iššifravimo programos darbo rezultatas, jei joje pateiksime šiek tiek pakeistą užšifruotą tekstą, pavyzdžiui, tekste bus pakeistas vienas simbolis: dbb6 → dbb8

- ❑ Išsiaiškinkite, kas vyks iššifruojant, jei pakeisite programos kitų dalių užšifravimo informaciją:
  - key
  - tag\_hex
  - nonce\_hex

<...>

```
def main():  
    # Pateikti duomenys  
    key = b"ThisIsASecretKey_32bytesLongAbCd"  
    ciphertext_hex =  
    "dbb6fd975fafc6282d8a27dd9841cc26bf67929926e764  
    8fa0b50fdc88bff2641ab5e68b7c447f310d527b6ff2df1f5  
    c4498c4be4ede13"  
    tag_hex = "265df1dfa03b68cd368605e2660c763d"  
    nonce_hex = "78db0191dcadb80bb318579ad3a535d3"
```

<...>

Čia pateiktas tik programos teksto fragmentas.



# Išbandykite simetrinio iššifravimo įrankį (5)

- ❑ Naudodami pateiktą simetrinio iššifravimo įrankį – programą **SIFR\_AESS.PY** – iššifruokite čia pateiktus užšifruotus pranešimus:

1.

Raktas: ThisIsASecretKey\_32bytesLongAbCd

Užšifruotas tekstas:

47b3ec03eacbe4aaddb22192c685f69da5cofab86d5b7cofb56e9coof1035ef59eb1146cbd44138d9e9fa69e7b3b8957a65ff9a3  
464a3eebb5762b9172e64844cfe380807c2baf7c4b11a23af2e25969a52e7275e2e1591dfd3cd0392c18c7db1c6397b69800

Tag: 692a8142327e1fd3de00bd76a1833931

Nonce: 3150360edd204dbcb610883b168b5958

2.

Raktas: ThisIsASecretKey\_32bytesLongAbCd

Užšifruotas tekstas:

a8cfed33a6ca1fd8bc16f5f712544a9c6350f20b5597495d46b4222b81c0cd7c4ead46e9811bcd7040d814348123cbc4718df4e03  
403106d7dc846e009755cd2943ed6a044761c410e6e88c5f2

Tag: 1d7fce75962fa02764f15b57769d097a

Nonce: 6900c5899fa76800ba7bff77e0fe6cb6

# Simetrinis šifravimas naudojant šifravimo raktą (1)

**Patarimas.** Prieš atliekant tolesnius veiksmus, patartina pakartotinai įdiegti / atnaujinti kriptografijos biblioteką. Tam komandų (CMD) eilutėje rašome **pip install --upgrade cryptography**

## 1. Naudodami Python SIM.PY programą, aplanke **c:\py** sukurkime simetrinį šifravimo raktą:

- nusikopijuokite čia pateiktą programos kodą į užrašinę ir įrašykite vardu **SIM.PY** į **C:** disko aplanką **py**;
- įvykdysite programą, komandų eilutėje nurodydami komandas:

```
cd c:\py  
python SIM.PY
```

# PROGRAMOS SIM.PY KODAS

```
# sim.py generuoja simetrinį raktą key.bin  
from cryptography.fernet import Fernet  
# Sugeneruojame simetrinį raktą  
key = Fernet.generate_key()  
# Išsaugome raktą į key.bin failą  
with open("key.bin", "wb") as key_file:  
    key_file.write(key)  
print(f"Raktas sugeneruotas ir išsaugotas į key.bin:  
{key}")
```

- aplanke **py** bus sukurtas **simetrinio šifravimo raktas** – failas **key.bin**.



# Simetrinis šifravimas naudojant šifravimo raktą (2)

2. Naudodami **Python** programą **Simetrinis\_e\_d.py** sukurtu simetrinio šifravimo raktu **key.bin** užšifruokite posakį:

*If at first you don't succeed, call it version 1.0.*

*(Jeigu iš pirmo karto nepasisėkė, pavadink tai versija 1.0. – Programuotojų išmintis)*

- Šį posakį įkelkite į programą „Užrašinė“ ir įrašykite į aplanką **c:\py** vardu, pavyzdžiui, **posakis.txt**.
- Į tą patį aplanką įrašykite ir simetrinio šifravimo / iššifravimo programą **Simetrinis\_e\_d.py** (jos kodas pateiktas kitoje skaidrėje)
- Komandų eilutėje nurodykite\*:  
**cd c:\py**  
**python simetrinis\_e\_d.py e key.bin posakis.txt posakis.bin**

\***Pastaba.** Jei įrašysite programą ir šifravimo raktą kitų vardu ir (arba) kitame aplanke, komandų eilutėje rašykite savo informaciją – nurodykite savo aplanką ir savo failų vardus.

# Simetrinis šifravimas

## naudojant šifravimo raktą (3)

### 3. Iššifruokite užšifruotą tekstą:

- Naudodami programą „Užrašinė“ atverkite failą **posakis.bin**. Kaip galvojate, ar įmanoma be rakto jį iššifruoti?
- Ištrinkite arba pervardykite aplanke **c:\py** pradinio teksto failą **posakis.txt**. Pabandykite jį atstatyti iš užšifruoto failo vykdydami komandas komandų (CMD) eilutėje:

**python simetrinis\_e\_d.py d key.bin posakis.bin posakis.txt**

- Perskaitykite iššifruotą tekstą, kurį programa įrašė į failą **posakis.txt**. Ar pastebėjote kokius nors neatikimus ar iškraipymus palyginus su pradiniu tekstu?

```
# python Simetrinis_e_d.py e|d key.bin Tekstas.txt Tekstas.bin
from cryptography.fernet import Fernet
import sys

def encrypt(key, plaintext):
    cipher_suite = Fernet(key)
    return cipher_suite.encrypt(plaintext)

def decrypt(key, ciphertext):
    cipher_suite = Fernet(key)
    return cipher_suite.decrypt(ciphertext)

def main():
    action = sys.argv[1]
    key_file = sys.argv[2]
    text_file = sys.argv[3]
    output_file = sys.argv[4]

    with open(key_file, 'rb') as f:
        key = f.read()

    with open(text_file, 'rb') as f:
        text = f.read()

    if action == 'e':
        encrypted_text = encrypt(key, text)
        with open(output_file, 'wb') as f:
            f.write(encrypted_text)
    elif action == 'd':
        decrypted_text = decrypt(key, text)
        with open(output_file, 'wb') as f:
            f.write(decrypted_text)
    else:
        print("Invalid command. Use 'e' for encryption or 'd' for decryption.")

if __name__ == '__main__':
    main()
```

# Simetrinis šifravimas naudojant šifravimo raktą (4)

- ❑ Naudodami programą **Simetrinis\_e\_d.py**, pabandykite iššifruoti čia pateiktą „labai slapta“ užšifruotą tekstą:

```
gAAAAABIFHwoMeVcPQwYOn2Cln9Uq41Kr2Gr_ND5e9_YKm2IVFqkL8MktROpvZdvJV9LoesjVWG6BTrUdN1EpSSBAIXQRgzOat4TMI  
5RclK8PjCoxXRElyosiOq8rGcvRLJqHJjH_WDYsFuTfOcWpSOY8zMqmlLZK7MzHXZwH8v_YQkEVstb5zuQ83OuhEBlgob8yqNfHnPodg  
EyYsjHxg_o3R116LLg==
```

- Pateiktą užšifruotą tekstą įkelkite į programą „Užrašinė“ ir įrašykite į **c:\py** vardu **slaptas\_tekstas.bin**.
- Į programą „Užrašinė“ įkelkite šį simetrinį raktą: **ucE\_arr6D5lyPcjNMU\_CuCUwl1PsfwBW7uEDBgVx2yU=** ir įrašykite į **c:\py** vardu **key.bin**.
- Iššifruokite šį tekstą naudodami programą **Simetrinis\_e\_d.py**, ir simetrinį raktą **key.bin**, komandų (CMD) eilutėje nurodydami:  
**python simetrinis\_e\_d.py d key.bin slaptas\_tekstas.bin slaptas\_tekstas.txt**
- Perskaitykite iššifruotą tekstą. Ar tikrai jį buvo galima laikyti labai slaptu? 😊

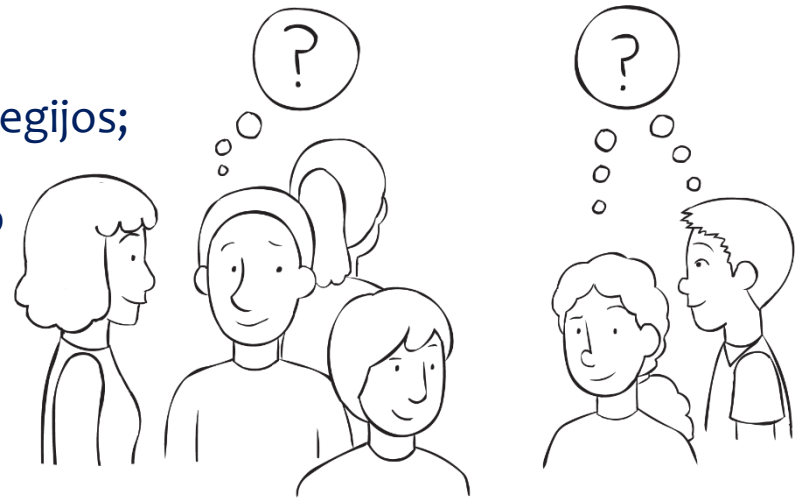
# Užduotis darbui grupėse: pranešimo užšifravimas ir iššifravimas **naudojant savo sukurtą šifravimo raktą (1)**

- ❑ Mokytojui padedant pasidalinkite į mažas grupes po 2–3 mokinius.
- ❑ Kiekvienas grupės narys:
  - naudodamas **Python** programą **SIM.PY**, sukurkite savo šifravimo raktą **key.bin**;
  - programa **Simetrinis\_e\_d.py** ir savo sukurtu šifravimo raktu **key.bin** užšifruokite norimą informaciją ir perduokite kitam grupės nariui (būtina perduoti ir šifravimo raktą – pagalvokite, kaip tai atlikti laikantis saugumo reikalavimų);
  - iššifruokite iš grupės draugo gautą informaciją (naudokite programą **Simetrinis\_e\_d.py** ir grupės draugo perduotą raktą **key.bin**).

# Užduotis darbui grupėse: pranešimo užšifravimas ir iššifravimas naudojant savo sukurtą šifravimo raktą (2)

❑ Grupelėse, po to ir visi kartu su mokytoju aptarkite:

- kaip pavyko atlikti užduotį;
- ar pakankamai saugios buvo jūsų pasirinktos šifravimo rakto perdavimo strategijos;
- ką galima būtų patobulinti siekiant didesnio užšifruotos informacijos saugumo.



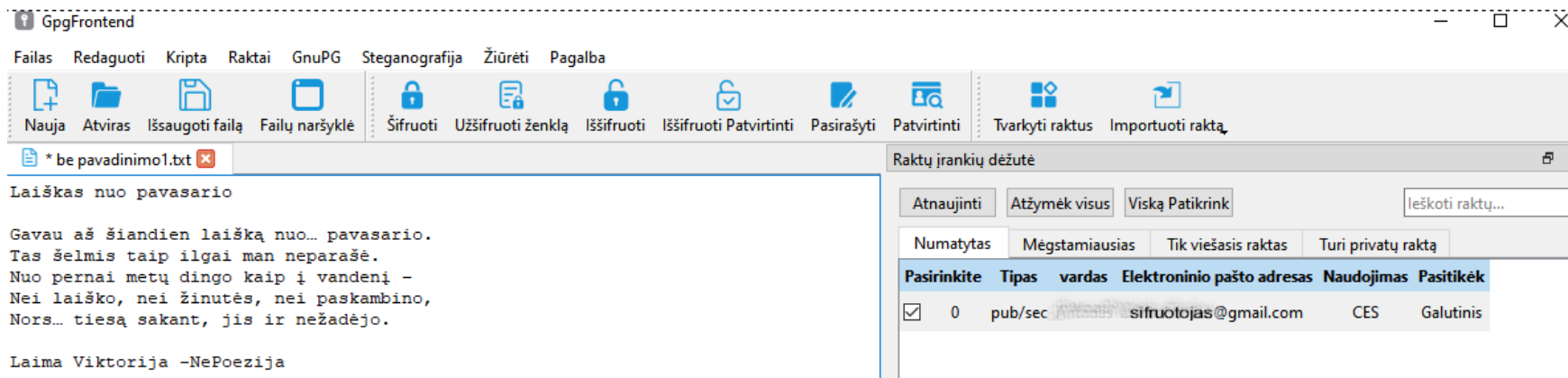
# Profesionalioje praktikoje naudojamos simetrinės kriptografijos sistemos pavyzdys (1)

- ❑ **GpgFrontend** yra kelių platformų šifravimo įrankis, kuris atitinka **OpenPGP standartą**. Jos tikslas – supaprastinti OpenPGP naudojimą, taip suteikiant galimybę daugiau asmenų apsaugoti savo privatumą.
  - **OpenPGP yra duomenų šifravimo ir iššifravimo standartas**, kurį palaiko GpgFrontend.
  - GpgFrontend labai gerai dokumentuota programa.
  - Programos svetainėje randama patarimų, kaip pasirašyti šifruotą pranešimą bei saugiai elgtis su raktais.

Atsisiųsti įrankį GpgFrontend: <https://www.gpgfrontend.bktus.com/#/downloads>  
Įrankio naudojimosi aprašas: <https://www.gpgfrontend.bktus.com/manual/encrypt-decrypt-text>

# Profesionalioje praktikoje naudojamos simetrinės kriptografijos sistemos pavyzdys (2)

Atsisiuntę ir įdiegę įrankį GpgFrontend, galite peržiūrėti ReadME instrukcijas (rodomas diegimo metu) ir pradėti ją naudoti atlikę vos kelis veiksmus.



GpgFrontend programos lango fragmentas

# SIMETRINIO ŠIFRAVIMO PRIVALUMAI IR TRŪKUMAI



# Simetrinis šifravimas – privalumai

- ❑ **Greitaveika.** Simetrinis šifravimas yra daug greitesnis nei asimetrinis. Tai reiškia, kad jūs galite užšifruoti ir iššifruoti daugiau informacijos per mažesnį laiką.
- ❑ **Mažiau resursų.** Simetriniai algoritmai paprastai reikalauja mažiau kompiuterio procesoriaus pajėgumo ir atminties, todėl jie yra patogūs naudoti įvairiose sistemose, net ir senesnėse ar mažesnio galingumo.
- ❑ **Paprastumas.** Nereikia rūpintis dviem raktų poromis kaip asimetriniame šifravime. Vienas raktas veikia tiek užšifravimui, tiek iššifravimui, tai palengvina procesą.

# Simetrinis šifravimas – trūkumai

- ❑ **Rakto pasidalinimo problema.** Didžiausias trūkumas yra būtinybė saugiai pasidalyti raktu tarp siuntėjo ir gavėjo. Jei kas nors nepageidaujamas gaus šį raktą, jis galės atšifruoti visą informaciją.
- ❑ **Rakto saugojimas.** Kadangi užšifravimo ir iššifravimo raktas yra tas pats, tiek išsiųstos, tiek gautos žinutės tampa pažeidžiamos, jei šis raktas prarastas ar pavogtas.
- ❑ **Mažesnis lankstumo lygis.** Simetrinio šifravimo algoritmai yra mažiau lankstūs pasirinkimų atžvilgiu, palyginti su asimetriniais. Jie dažniausiai yra tinkami tik duomenų šifravimui, bet ne identifikavimui ar elektroniniams parašams, kas puikiai atliekama naudojant asimetrinį šifravimą.

# ASIMETRINĒ KRIPTOGRAFIJA

# Asimetrinis šifravimas

- **Asimetrinio šifravimo sistema yra kriptografijos algoritmas**, kuriame naudojami du matematiškai susiję, bet skirtingi raktai – **viešasis raktas** ir **privatus raktas**.
  - **Viešasis raktas** yra prieinamas visiems ir gali būti **naudojamas** informacijos **užšifravimui** arba el. parašui tikrinti.
  - **Privatus raktas** yra saugomas paslapyje ir naudojamas informacijos iššifravimui arba el. parašo sukūrimo procese.
  - Viešasis ir privatusis raktai yra tarpusavyje susiję taip, **kad informacija, užšifruota viešuoju raktu, gali būti iššifruota tik atitinkamu privačiu raktu**. Tačiau naudojant šiuos raktus, negalima lengvai atkurti vieno ar kito iš jų – turint viešąjį raktą, negalima lengvai gauti privataus rakto.

Prisiminkite (arba atlikite) ir aptarkite užduotį, demonstruojančią, kaip vyksta keitimasis informacija naudojant asimetrinio šifravimo metodą (užduotis aprašyta 9 – 10 klasės metodinėje medžiagoje – pateiktyje ir užduočių faile).

## Užduotis (veikla), demonstruojanti, kaip vyksta asimetrinis šifravimas

### Aprašomos veiklos esmė

- Parodoma, kad informacijai šifruoti asimetriniu šifravimu atliekami du skirtingi veiksmai: vienas, kuris atliekamas (prieš išsiunčiant) informacijai užrašyti (užšifruoti), ir kitas, kuris atliekamas (gavus slantą) informacijai atkurti (iššifruoti), ir kad šie veiksmai atliekami naudojant atitinkamus skirtingus raktus – viešąjį ir privatų.
- Aprašant šios užduoties veiklas, pateikiamas pranešimo užšifravimo bei iššifravimo realaus proceso žingsnių imitavimas.
- Šis pavyzdys yra paprastas, bet jis leidžia suprasti asimetrinio šifravimo esmę: galimybę saugiai perduoti informaciją naudojant skirtingus raktus užšifravimui ir iššifravimui.

### Priemonės

- 3 dėžutės su dviem užraktais („viešuoju“ ir „privatu“).
- „privatus“ + „viešasis“ raktai kiekvienai dėžutei.
- Etiketės, žymekliai arba kitos priemonės dėžučių ir raktų žymėjimui.
- Lapeliai užraktams, raktymo priemonės.



### Priemonių alternatyvos

Neturint dėžučių, galima naudoti simbolines „dėžutes“ (tai gali būti vokai, kartoninės dėžutės arba suvėlioti popieriaus lapai) ir simbolines „raktus“ (pavyzdžiui, „viešasis raktas“ ir „privatus raktas“ gali būti nudaikti spalvoti ir (arba) atitinkamai pažymėti lapeliai).



### Veikla (1)

#### Pasiruošimas:

- Sudomros trys mokinių grupės, kiekviena grupė gali sugalvoti savo grupės pavadinimą. Mes paprastumo dėlei grupes pavadinsime A, B ir C.
- Kiekviena iš trijų grupių gauna grupės vardą (A, arba B, arba C) pažymėtą dėžutę bei grupės vardą (A, arba B, arba C) pažymėtus raktus – „viešuosius“ ir vieną „privatų“ (jį reikia dar papildomai pažymėti, pavyzdžiui, grupės vardu „A“ arba „B“ arba „C“).

### Veikla (2)

#### Raktų mainai:

- Grupė A perduoda grupėms B ir C po vieną savo „viešąjį“ raktą – šis raktas kitų grupių bus naudojamas tik tam, kad grupės B ir C grupės A dėžutę įdėtų užrašytą „paslaptį“ („užšifruotą“ pranešimą). „Privatų“ raktą (A) grupė A pasilieka sau.
- Tokius pat veiksmus atlieka ir grupės B bei C – perduoda savo „viešuosius“ raktus kitoms grupėms, sau palikdami „privatų“ raktą ir vieną „viešąjį“.

### Veikla (3)

#### „Paslapčių“ įdėjimas ir užrašymas (užšifravimas):

- Grupė A atidaro grupių B ir C dėžutes naudodamiesi atitinkamais tų grupių „viešaisiais“ raktais, įdėda „paslaptį“ (informaciją, kurią užšifruojama ir perduodama kitai grupei, pavyzdžiui, pranešimą „Ši ketvirtadienį mokykloje vyks geografinis naktis. Renkames 20 valandą sporto salėje. Nepamirškite pasiimti vandens ir maisto.“) ir vėl užrakina („užšifruoja“).
- Taip pat elgiasi ir grupės B bei C.

### Veikla (4)

#### „Paslapčių“ atrakinimas (iššifravimas):

- Grupė A naudoja savo „privatų“ raktą (raktą, kurio neatidavė kitoms grupėms, o pasilikė sau) „paslaptį“, kad atrinktų savo dėžutę ir raktų kitų grupių įdėtą (atsiųstą), „paslaptį“ („iššifruoja“ perduotą pranešimą).
- Grupės B bei C daro tą patį.

### Veikla (5)

#### Aptarimas:

- Kiekviena grupė perkaito atsiųstas „iššifruotas paslaptis“ ir visi kartu su mokytoju aptaria, kokie yra atliktos veiklos ir asimetrinio šifravimo žingsnių atitikimai (pavyzdžiui, vienu – viešuoju – raktu užrakiname („užšifruojame“), kitu – privatu – raktu atrakiname („iššifruojame“)).



Užduotis su išsamiau aprašymu pateikta metodinės medžiagos 9–10 klasių pateikties 29–37 skaidrėse.

# Viešojo ir privataus rakto sąvokos

## Viešasis raktas (Public Key)

Viešasis raktas yra asinchroninio šifravimo kriptografinės raktų poros dalis, kuri yra viešai prieinama. Šis raktas naudojamas šifruoti duomenis arba patikrinti elektroninio parašo autentiškumą. Kiekvienas, turintis viešąjį raktą, gali šifruoti duomenis taip, kad tik asmuo, turintis atitinkamą privatų raktą, galėtų juos iššifruoti.

## Privatus raktas (Private Key)

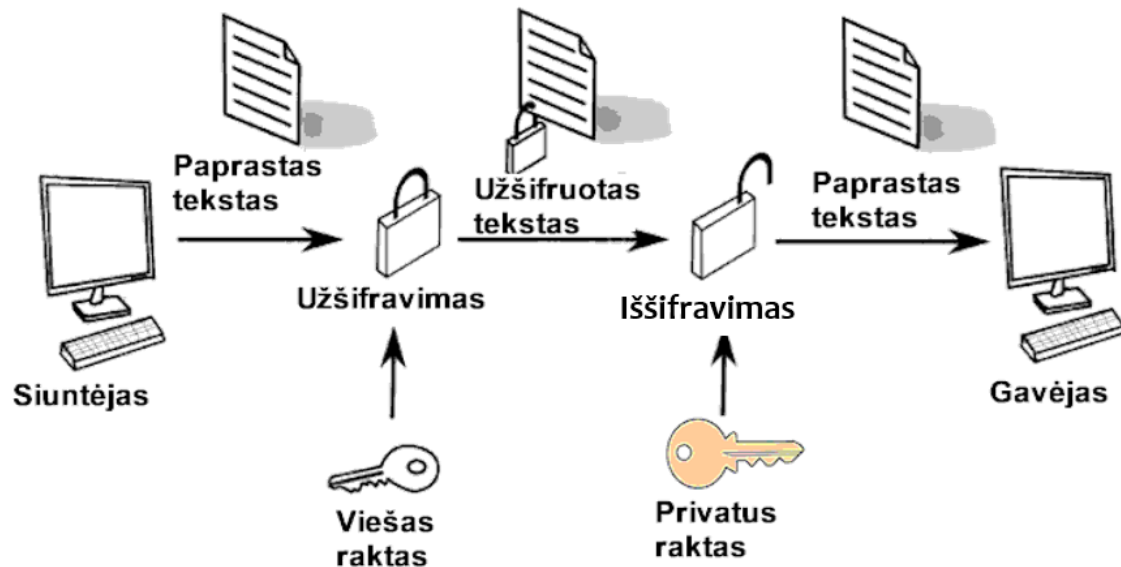
Privatusis raktas yra asinchroninio šifravimo kriptografinės raktų poros dalis, kuri saugoma kaip paslaptis. Šis raktas naudojamas iššifruoti duomenis, kurie buvo užšifruoti naudojant atitinkamą viešąjį raktą, arba sukurti elektroninį parašą. Privačiojo rakto saugumas yra labai svarbus, nes su juo galima atkurti visus užšifruotus viešuoju raktu duomenis ir atlikti kitas saugumo kritines operacijas.

# Sertifikato sąvoka

## Sertifikatas (Digital Certificate)

Sertifikatas yra elektroninis dokumentas, kuriame patvirtinama viešojo rakto ir identifikatoriaus (pvz., vartotojo vardo ar organizacijos pavadinimo) sąsaja. Sertifikatai išduodami ir patvirtinami pasitikėta trečiaja šalimi, vadinama sertifikavimo centru (CA - Certificate Authority). Sertifikatai naudojami patikrinti asmenų, sistemų ar tarnybų tapatybę internete.

# Asimetrinio šifravimo schema



**Užduotis.** Yra sakoma, kad simetrinio šifravimo atveju sauga ypatingai priklauso nuo šifravimo rakto saugojimo ir jo saugaus perdavimo komunikacijos dalyviui. Pakomentuokite, kodėl šios galimos simetrinio šifravimo spragos iš esmės dingsta naudojant *asimetrinį šifravimą*.



- ❑ Asimetrinio šifravimo metodai yra ypač naudingi saugiam duomenų perdavimui per nesaugius tinklus, tokius kaip internetas, taip pat tapatybės patvirtinimui, elektroniniams parašams ir kitoms panašioms operacijoms.
- ❑ Asimetrinė kriptografija yra plačiai naudojama įvairiose pramonės ir technologijų srityse dėl jos gebėjimo užtikrinti aukštą saugumo lygį ir konfidencialumą.

# Asimetrinė kriptografija – taikymas (1)

- ❑ **Interneto saugumas (TLS/SSL protokolai)** – užtikrina saugų informacijos perdavimą tarp interneto naršyklių ir serverių.
- ❑ **Tinklo technologijos (SSH – Secure Shell)** – suteikia saugią prieigą prie nuotolinių serverių.
- ❑ **VPN (VPT – Virtualūs privatūs tinklai)** – naudoja asimetrinę kriptografiją saugiam duomenų perdavimui tarp kliento ir serverio.
- ❑ **Elektroniniai laiškai ir komunikacija (PGP/GPG – Pretty Good Privacy/GNU Privacy Guard)** – šifravimas saugiai perduodant elektronines žinutes.

# Asimetrinė kriptografija – taikymas (2)

- ❑ **Duomenų saugojimas, debesijos duomenų saugojimo** paslaugos – kai kurios debesijos paslaugos naudoja asimetrinę kriptografiją failų šifravimui ir prieigos kontrolės mechanizams.
- ❑ **Skaitmeniniai parašai.**
- ❑ **E. dokumentų patvirtinimas** – dokumentų ir sutarčių skaitmeniniai parašai, patvirtinami naudojant asimetrinę kriptografiją.
- ❑ **Finansinės paslaugos.**
- ❑ **Mokėjimų sistemos** – asimetrinė kriptografija užtikrina, kad piniginės operacijos atliekamos saugiai.

# Asimetrinė kriptografija – taikymas (3)

- ❑ **Blockchain ir kriptovaliutos** – naudoja asimetrinę kriptografiją transakcijų autentifikavimui.
- ❑ **Identifikacija ir autentifikacija.**
- ❑ **Skaitmeniniai sertifikatai** – asimetrinė kriptografija naudojama elektroninėse tapatybės kortelėse, vairuotojo pažymėjimuose ir kituose tapatybės dokumentuose.
- ❑ **Įrenginių autentifikacija ir saugus duomenų perdavimas** – dėl ribotų skaičiavimo galimybių ypač svarbu naudoti efektyvius asimetrinės kriptografijos algoritmus.

# Asimetrinė kriptografija – taikymas (4)

- ❑ **IoT (Internet of Things)** – daiktų internetas.
- ❑ **Mašininis mokymasis ir duomenų analizė.**
- ❑ **Homomorfinė kriptografija** – leidžia atlikti skaičiavimus su šifruotais duomenimis, kuriuos vėliau galima dešifruoti ir gauti teisingus rezultatus.
- ❑ **Valdžios ir karinės institucijos.**
- ❑ **Ryšių šifravimas, skaitmeniniai sertifikatai** – naudojami saugiai komunikacijai ir duomenų laikymui.
- ❑ **Ir kitos taikymo sritys.**

Programavimo kalbos Python panaudojimo pavyzdys kuriant programą, kuri naudoja asimetrinio RSA (Rivest–Shamir–Adleman) šifravimo biblioteką, kad užšifruotų ir iššifruotų pateiktą tekstą.

# Šifravimo programos SIFR\_RSA.PY darbo rezultatas

(programos SIFR\_RSA.PY kodas pateiktas kitoje skaidrėje)

```
C:\Users\ioab>python3.11 c:\PY\SIFR_RSA.PY
```

```
Viešasis raktas:
```

```
-----BEGIN PUBLIC KEY-----
```

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAg8DaLFeZymJm7xrQbhY  
WcGTMD3DhVgf9K6eSYcolch5TevYNo6aMaNaHZkmstm1oMNA1GoF5i373K19UG  
duS8renYK4mL7H01adKR2q1G1QVa/66011sVQMEYJbYR0/1VtNk/w+ARbMZUEgG2  
Q0+Gg/kjI1drwA/jwipVFqbyD0cZJexjtdOeUNoz/p7o+TM7IyXMuqQIdKYXq+Qh  
Yhw80V0h55RN1w4tptCDSZEPm+vS+HtT4bnfKRXunSCEHICeBwbZ+QEJgXFz1Ee/  
jIeG11dpdhmcZpaxm5e2yU14DPAEHohZ1p5jrhxfDRTFwM1QrtXIZ/4ffqbo7U+6  
PwIDAQAB
```

```
-----END PUBLIC KEY-----
```

```
Privatusis raktas:
```

```
-----BEGIN RSA PRIVATE KEY-----
```

```
MIIEowIBAAKCAQEAg8DaLFeZymJm7xrQbhYWcGTMD3DhVgf9K6eSYcolch5Tev  
YNo6aMaNaHZkmstm1oMNA1GoF5i373K19UGduS8renYK4mL7H01adKR2q1G1QVa  
/66011sVQMEYJbYR0/1VtNk/w+ARbMZUEgG2Q0+Gg/kjI1drwA/jwipVFqbyD0cZ  
JexjtdOeUNoz/p7o+TM7IyXMuqQIdKYXq+QhYhw80V0h55RN1w4tptCDSZEPm+vS  
+HtT4bnfKRXunSCEHICeBwbZ+QEJgXFz1Ee/jIeG11dpdhmcZpaxm5e2yU14DPAE  
HohZ1p5jrhxfDRTFwM1QrtXIZ/4ffqbo7U+6PwIDAQABAB/MXpWiA1StKM22  
q7M3KyT/W8hze2nAc3EOVJW1c0SiZMAGJKBCE3T0n97JE2z0DMbzmpYH6J3F  
eCHUiuYnj1xemM2BvjnoQ47yZQax69XcwJ8BhrczcgETC0rH5Ly+XrZsxdEjKyf/  
VbZF4SRuX+d46g17st2x1+hhxvyTL7B+JYycQvakWJo1HfKsQ0jbaIEc+XQcOI8Z  
X8+Zr7sEL3j2W05wsyF1Jxc0AIO6PgTbDk80FsY3a5vWkTtXg4lNlMolnapCKH  
5uBQ0jbrj7funsEGRa4d66pVFM46ApC1JCqf/HHJt+D0BuxsHg541xgNl+R7Jc  
Q19ozgEGcYEaZQW2Ip4UviTwmcen9kn3j2npZmb0PdanFs9E4HeqmxVLQXNE7Ur  
eh08eUQv+151832+Xsr5edd1p36BF1XhZQX0HndBMYhw+/FCLV8e6FFZwJlT868u  
WN5JnzRjvjZ8hrHDMCynJdKqWEX4sSol1WtpK51s9nsFQT3RmIK3cCEgYEAtmAW  
M458aVpaOxAE86ng8MuQqXrF41SwxV1T1wYjYe/mlI6/1Re6+X5xGwTnwzUuxxy  
tJ5daobgVrao1pfnBkPlbyVTc6UYvLR1WlwM9SrDmFOBA6ci1hPmH4hgnZ17WdTD  
8M3j517r8CazTbQRLicaUicKv9LP58CEIS2K1/8CgVBXh/S22t6721PF+pPPj50K  
Wm6+2lHYhFdmG3179ppmEFJduZXNaC/jssFo1J/XOKZuX+UC+415/LYSgobd01U9  
SeT0FAena1Uyrrh1NKAZHEVA6VjD/f09Aqcas3dn53QXHZXD34yNlV/kyuFMYj4  
caVtBXBgamfMoY4g11uVAQKBgQCCxuZEtDEMjwJ5/4q9e5zcrucyE3a9NgJ5uNP  
//115sej3HvP8epCzG0030n5bLkkCE0CBmmX2SGdL8oNspEvxeIE+eB5515uY4  
FHgKOK8nD19Y3V/c6d6Bx5ZBN1KLMMPQ2Tg3dVMMUQUPUjwQqoGuRmEd7xTY6  
YGdy2uK8Bgw7yO4gSV688tHUISGaou94n1kY7ktsDVcqh780t8DTYz1dJdEcmgw0  
M9dLET3Cn/c8C33G9fd+bvMrc1wVtBT+Ry23GPLUdahyNgZ4Mik4GUJ8Juaaj1vb  
C4HGdy0KprkgwPUuLrEqeHaJu0Twrh0J2FWG2a7LLJMDZ1+9hVlQ
```

```
-----END RSA PRIVATE KEY-----
```

```
Užšifruota žinutė: b'\x1K57p8\xacR&\x88\xbd%\x912\x2e\xba\x1b\x9d\x9f\x8d\xeeZ~\xc35\xff\x3\x1\x8bQ\x99\x8f\x95\xda\xed\xef1\x6\xec\x0f\x12J\xea\x0\xfe \xc3e\x8d\x2q\x9a\x83t\x97 \x13Q\x3fN\x8a\x92P\x89E\xac\x0f\x9f\x6\x1f\x6  
0DNK;\x1a\x90\x8f\x0e\xdd2\tn\x99\x88\x2M\xdf:\xb6c\x40@V\xbb\x97\x8e\x86\x4d9d\x19FDKa\x2c\x11?' \xdadAn\x070\x8d\x10\xaf\x23R42v\t%\x98\xaa\x8b\x8e3\x08\x9a5\x2f4\x86\x9f9\x9bq\x17\x25\x2f7\x98?b,\xdd\x05\x1e13)\x18\x07\x29Gg063V\xafQ\x  
91J\x2a2\x8b:\x97\x23d\x2e2\x2a6\x2c1\x21\x28c\x80\x8e3\x88Bum3\xbb0i+\x95\x0f5\xaf#%,\xbf:[\xb9~\u0x2\x8b\xfa\x8b\x180\xba\xbb\xaf2\x3Q\xde\x9a\x9e\x24\x99\x80V\x26\xbcw\x25\x2a0\x2c6&UK\x24\x110\x1ca(\x858*\x9f_1\x2c0u\x21%>\x26\x91\x2b2\x16q
```

# Asimetrinis šifravimas naudojant Python kalba sudarytą programą SIFR\_RSA.PY

(prieš naudojant įdiekite biblioteką – komandų eilutėje nurodykite `pip install cryptography`)

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import rsa,
padding
```

```
# Generuojame RSA privatuji ir viešąjį raktą
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
    backend=default_backend()
)
public_key = private_key.public_key()

# Eksportuojame viešąjį raktą PEM formatu
public_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)
print("Viešasis raktas:")
print(public_pem.decode("utf-8"))
```

```
# Eksportuojame privatuji raktą PEM formatu
private_pem = private_key.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.TraditionalOpenSSL,
    encryption_algorithm=serialization.NoEncryption()
)
```

```
print("Privatusis raktas:")
print(private_pem.decode("utf-8"))
```

```
# Užšifruojame žinutę su viešuoju raktu
original_message = b"Svarbi informacija"
ciphertext = public_key.encrypt(
    original_message,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print("Užšifruota žinutė:", ciphertext)
```

```
# Iššifruojame žinutę su privačiuoju raktu
decrypted_message = private_key.decrypt(
    ciphertext,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print("Iššifruota žinutė:", decrypted_message.decode('utf-8'))
```



# Kurį pasirinkti?

- ❑ **Simetrinis šifravimas** yra greitas ir paprastas, bet kyla problemų, kai reikia saugiai pasidalyti ir saugoti raktus.
- ❑ **Asimetrinis šifravimas** šias problemas sprendžia, bet yra lėtesnis ir reikalauja daugiau kompiuterio resursų.

\*\*\*\*\*

**Kiekvienu atveju reikia pasirinkti labiausiai tinkamą šifravimo strategiją arba šifruojant informaciją derinti abi galimybes – simetrinį šifravimą su asimetriniu.**

# Simetrinio ir asimetrinio šifravimo derinimo pavyzdys (1)

- ❑ Kadangi asimetrinė kriptografija yra daug lėtesnė nei simetrinė kriptografija, ji retai naudojama dideliems duomenų kiekiams tiesiogiai užšifruoti.
- ❑ Vietoj to, asimetrinis šifravimas paprastai naudojamas tokiu būdu simetriniams raktams užšifruoti.

## Simetrinio ir asimetrinio šifravimo derinimo pavyzdys (2)

- ❑ Agnė ir Benas dažnai susirašinėja rengdami bendrą projektą.
- ❑ Išnagrinėkite Agnės ir Beno, kurie keičiasi informacija, susitarimus ir veiksmus:
  - Agnė reikalauja, kad Benas el. paštu siųstų jai tik užšifruotas žinutes.
  - Ji sukuria privataus/viešojo rakto porą, saugo savo privatų raktą paslapyje ir paskelbia savo viešąjį raktą.
  - Benas turi žinutę, kurią nori nusiųsti Agnei.
  - Jis sukuria naują simetrinį raktą ir šį raktą panaudoja, kad užšifruotų savo pranešimą Agnei.
  - Po to Benas naudoja Agnės viešąjį raktą, kad užšifruotų savo simetrinį raktą.

(tęsinys kitoje skaidrėje) →

## Simetrinio ir asimetrinio šifravimo derinimo pavyzdys (3)

- Benas siunčia užšifruotą pranešimą ir užšifruotą simetrinį raktą Agnei (toks raktas vadinamas **suvyniotu** arba **apgaubtu** raktu).
  - Agnė naudoja savo privatų raktą (iš privačios/viešos poros), kad iššifruotų simetrinį Beno raktą.
  - Po to ji naudoja simetrinį Beno raktą, kad iššifruotų pranešimą.
- Kaip manote, ar pakankamai saugiai buvo perduotas Agnei Beno sukurtas simetrinis šifravimo raktas?
  - Aptarkite tai bendroje grupės diskusijoje.

(adaptuota pagal <https://learn.microsoft.com/>)