



Co-funded by
the European Union

Internet and Computer Security

Amel Toroman, Ermin Bajramović
UNBI



UNIVERSITY OF LJUBLJANA
Faculty of Electrical Engineering



University of Pristina
Kosovska Mitrovica



Internet and computer security

INTRODUCTION

The internet and computer security are crucial factors for the safety of the digital world we live in.

As technology advances, so does the need for adequate protection of computer systems, data, and user information on the internet.

In this course, through a detailed theoretical foundation and practical examples, key topics such as internet security, security threats, protection methods, and cryptography - the cornerstone of data and communication security - will be covered.



Internet and computer security

INTRODUCTION

Special attention will be given to the application of Python in implementing protective methods, as well as the analysis and implementation of various cryptosystems, such as symmetric and asymmetric cryptography, hash functions, and random number generators.

The course is designed to enable participants to understand the theoretical foundations and practical skills required for managing security in a digital environment.



Internet and computer security

INTERNET

The **internet** is a global network of computers that enables communication and data exchange using various technologies and protocols.

It originated as a result of the development of computer networks and today forms the foundation of the modern digital world. Billions of devices are connected through the internet, and its role is present in almost every aspect of human life - from education and business to entertainment and social interactions.



Internet and computer security

INTERNET

The internet operates based on a set of rules and standardized protocols that enable communication between computers. Key elements of the internet include:

- IP addresses (Internet Protocol Address)
- DNS (Domain Name System)
- Data transfer protocols
- Internet services

However, the use of the internet also carries certain security risks. Therefore, it is essential to understand the protection methods that enable safe internet usage.



Internet and computer security

Methods of protecting computers and users on the Internet

Malveri (Malware) - virusi, trojanci, ransomware su zlonamjerni softveri dizajnirani s ciljem narušavanja sigurnosti korisnika.

Najčešće vrste malvera su:

- **Virusi** - Računarski programi koji se prikače na druge legitimne programe i izvršavaju štetne radnje.
- **Trojanci** - Maliciozni softveri koji se prikrivaju kao korisni programi, ali zapravo omogućavaju hakerima pristup sistemu.
- **Ransomware** - Malver koji šifrira podatke na računaru i zahtijeva otkupninu za njihovo vraćanje.



Internet and computer security

Methods of protecting computers and users on the Internet

Phishing and Social Engineering

Phishing attacks are based on deceiving users through fake emails or websites that mimic legitimate services. The goal is to trick users into entering their confidential information, such as passwords and credit card numbers.

Network Attacks (DDoS, MITM):

- **DDoS Attacks** - Attackers use networks of infected computers (**botnets**) to overwhelm servers with a large number of requests, rendering them inaccessible.
- **MITM (Man-in-the-Middle) Attacks**- Attackers intercept communication between a user and a web service, allowing them to steal data or manipulate information.

Internet and computer security

Methods of Protection on the Internet

Antivirus Software and Firewall

Antivirus programs are essential for protecting against malware, while a **firewall** filters network traffic and prevents unauthorized access.

EXAMPLE: Activating Windows Defender

Settings → Update & Security → Windows Security → Virus & threat protection

EXAMPLE: Activating Gmail 2FA

Google Account Settings → Security → 2-Step Verification

Internet and computer security

Methods of Protection on the Internet

Data Encryption and VPN

VPN services provide additional security by encrypting internet traffic and hiding the user's IP address.

EXAMPLE: Installing Windscribe VPN

<https://windscribe.com/>



Internet and computer security

Example of Internet Protection Using Python

Checking Website Security Using a Python Script

The following Python script uses the VirusTotal API to check if a website is malicious.

```
import requests

API_KEY = "YOUR_VIRUSTOTAL_API_KEY"
url_to_check = "http://example.com"

def check_url_safety(url):
    params = {"apikey": API_KEY, "resource": url}
    response = requests.get("https://www.virustotal.com/vtapi/v2/url/report",
    params=params)

    if response.status_code == 200:
        result = response.json()
        if result["positives"] > 0:
            print(f"Upozorenje: {url} je označen kao nesiguran!")
        else:
            print(f"{url} je siguran za posjetu.")
    else:
        print("Greška pri provjeri URL-a.")

check_url_safety(url_to_check)
```



Internet and computer security

Example of Internet Protection Using Python

Running the Python Script

This script sends a URL to VirusTotal and checks whether it is flagged as malicious.

Initially, you need to sign up on the VirusTotal website, copy the API key for verification, and replace it in the code with YOUR_VIRUSTOTAL_API_KEY.

After that, for testing, set the URL you want to check at ``url_to_check = "http://example.com"``.

The next step is to open the terminal and run the file using the following command:

```
python NameFile.py
```

Internet and computer security

Example of Internet Protection Using Python

If the website is reported as malicious, the following message will appear, as shown in the image.

```
C:\Users\AzraKT\Desktop>python zastita.py  
Warning: https://web.tfb.unbi.ba has been marked as unsafe!
```

If the website is not reported as malicious, the following message will appear, as shown in the image.

```
C:\Users\AzraKT\Desktop>python zastita.py  
https://www.bhtelecom.ba is safe to visit.
```



Internet and computer security

COMPUTER SECURITY AND DATA PROTECTION

With the development of mathematical sciences, technology, and computing, complex algorithms emerged that involved intricate mathematical calculations.

Encryption, access control, and document protection behind so-called firewalls are some of the common techniques used to protect sensitive information.

Along with these methods, digital signatures and digital watermarks are also utilized.

Essentially, attacks can be defined as actions aimed at compromising the security of information, computer systems, and networks.



Internet and computer security

COMPUTER SECURITY AND DATA PROTECTION

When discussing specific examples, significant threats come from malicious software, which includes:

- 1) Viruses,
- 2) Worms,
- 3) Trojan horses,
- 4) Spyware and adware,
- 5) Tools for gaining administrative privileges on a system (rootkits), etc.

Some of the most notable software protection methods include:

- 1) Encryption,
- 2) Use of watermarks, ciphers, and cryptographic hashes,
- 3) Utilization of digital signatures.



Internet and computer security

CRYPTOGRAPHY

Cryptography is a scientific discipline that studies methods for sending messages in such a way that only the intended recipient can read them.

Effective communication has been a crucial factor for kings, queens, and military leaders for centuries, due to the awareness of the consequences that could result if their messages fell into the wrong (enemy) hands.

In this way, forbidden, highly important, and valuable secrets could be revealed.

This danger prompted the development of ciphers and codes, which can be defined as means by which a message can only be read by the person it is intended for.



Internet and computer security

Basic Concepts in Cryptography

Cryptography is a scientific discipline that studies methods for sending messages in forms that are readable only to those for whom they are intended.

The word **cryptography** is of Greek origin and could be literally translated as "secret writing."

The goal of cryptography is to enable uninterrupted communication between the sender and the receiver of a message over an insecure communication channel, so that a third party cannot understand their messages.



Internet and computer security

Basic Concepts in Cryptography

A cryptographic algorithm or cipher is a mathematical function used for encryption and decryption.

Generally, it involves two functions: one for encryption and the other for decryption. These functions map the basic elements of plaintext (typically letters, bits, or groups of letters or bits) to the corresponding elements of ciphertext and vice versa.

The functions are selected from a specific family of functions depending on the key. The set of all possible key values is called the **key space**.




Internet and computer security

Basic Concepts in Cryptography

Encryption is a process applied on the sender's side. It transforms plaintext into ciphertext or encoded text using the appropriate key.

Therefore, encryption is divided into **ciphering** and **coding**. In ciphering, the basic unit is a single letter, a pair of letters (bigram), or sometimes even more letters (poligram), while in coding, the basic unit is a word or a set of words. As a result, there is no clear theoretical boundary between codes and ciphers.

Ciphering is a process that is further divided into **transposition** (where each letter retains its identity but changes its position) and **substitution** (where each letter retains its position but changes its identity).



Internet and computer security

Basic Concepts in Cryptography

Decryption is the process by which ciphertext or encoded text is transformed back into plaintext using a previously known key and algorithm.

It is applied on the recipient's side of the message. Therefore, this process is legal, as the recipient is communicating with the sender.

The result of this process is called the **decrypted text**.



Internet and computer security

Basic Concepts in Cryptography

A cryptosystem consists of a cryptographic algorithm or cipher, along with all possible plaintexts, ciphertexts, and keys. The criteria for classifying cryptosystems are as follows:

1) Type of operations used in encryption

- a) *Substitution ciphers* - Each element in the plaintext is replaced with another element.
- b) *Transposition ciphers* - The positions of elements in the plaintext are rearranged, but the elements themselves remain the same.

2) Method of processing plaintext

- a) *Block ciphers* - The plaintext is processed one block at a time, using the same key for each block.
- b) *Stream ciphers* - The elements of the plaintext are processed one by one, using a series of keys (called a keystream) that is generated in parallel.

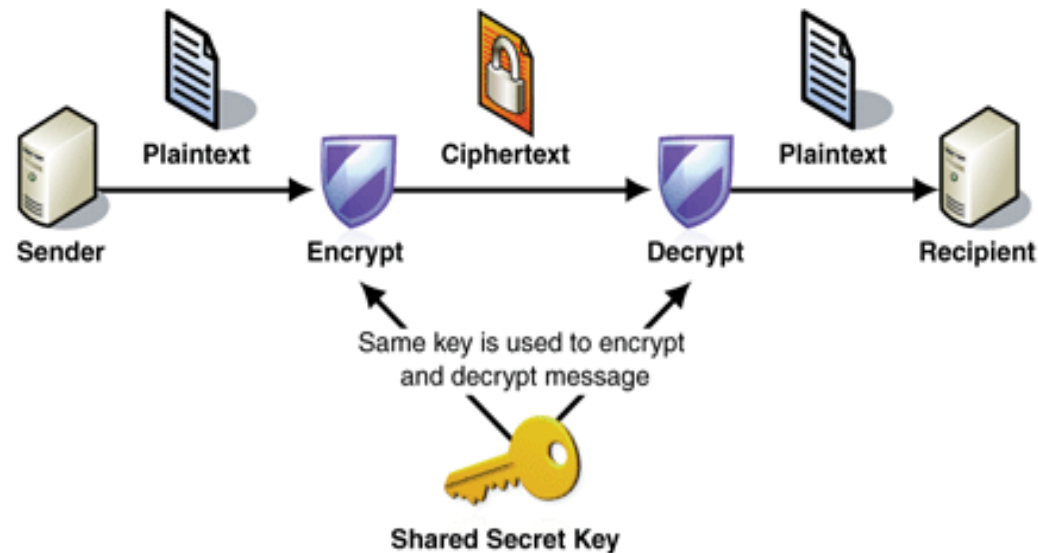
3) Confidentiality and publicness of keys:

- a) *Symmetric key cryptosystems* - Both encryption and decryption use the same secret key.
- b) *Asymmetric key cryptosystems* - A public key is used for encryption, and a private key is used for decryption.

Internet and computer security

Cryptosystems with a Secret Key

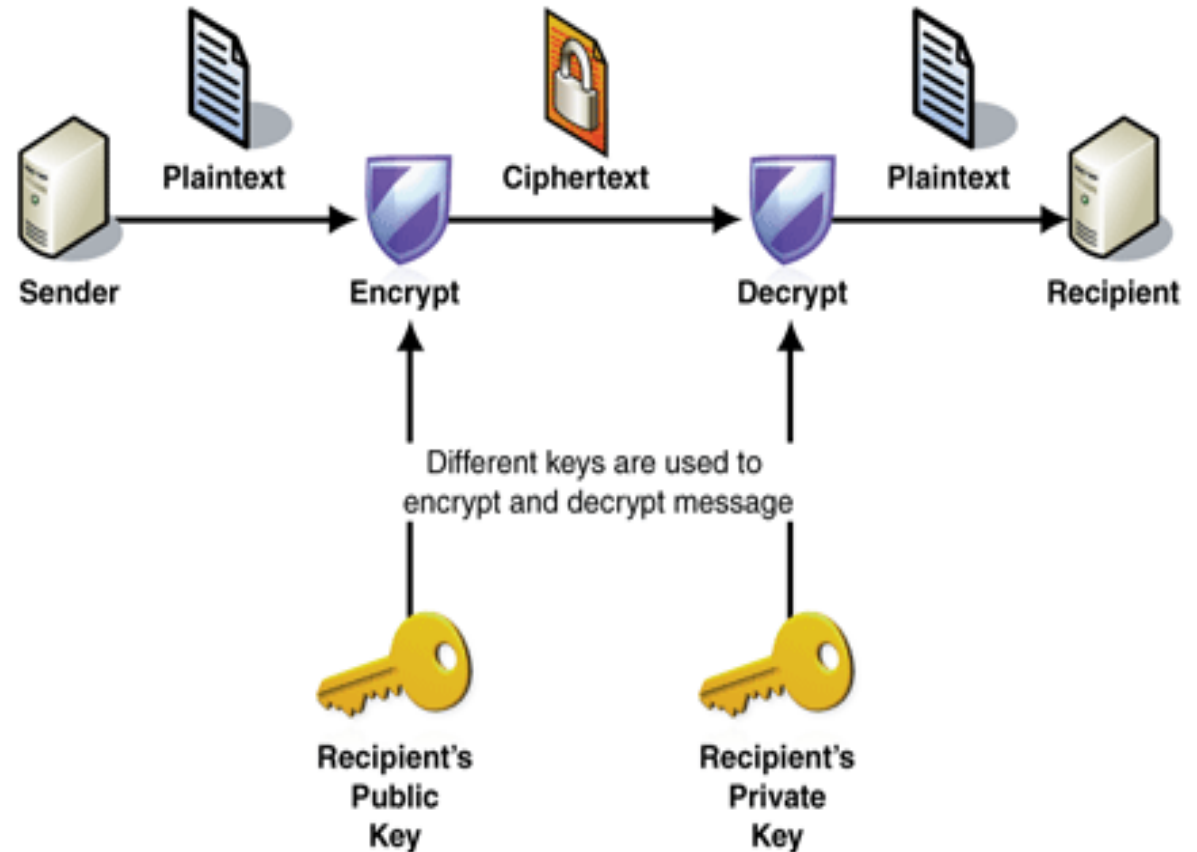
In symmetric, conventional cryptosystems (i.e., cryptosystems with a secret key), the sender and the receiver permanently choose a key K that they use to generate functions for encryption and decryption. In this case, the functions are identical, which makes them vulnerable to cryptanalysis or "breaking."



Internet and computer security

Cryptosystems with a Public Key

In public key cryptosystems (asymmetric cryptosystems), it is computationally infeasible, within a reasonable time, to determine the decryption key (despite the fact that the encryption key is public). Specifically, anyone can encrypt a message using the public key, but only the person who possesses the corresponding decryption key (the private or secret key) can decrypt the message.



Internet and computer security

Types of Cryptographic Algorithms

Cryptographic algorithms are divided into:

1. Symmetric algorithms

The same key is used for both encryption and decryption (AES, DES, 3DES).

2. Asymmetric algorithms

A public key is used for encryption and a private key for decryption (RSA, ECC).

3. Hash functions

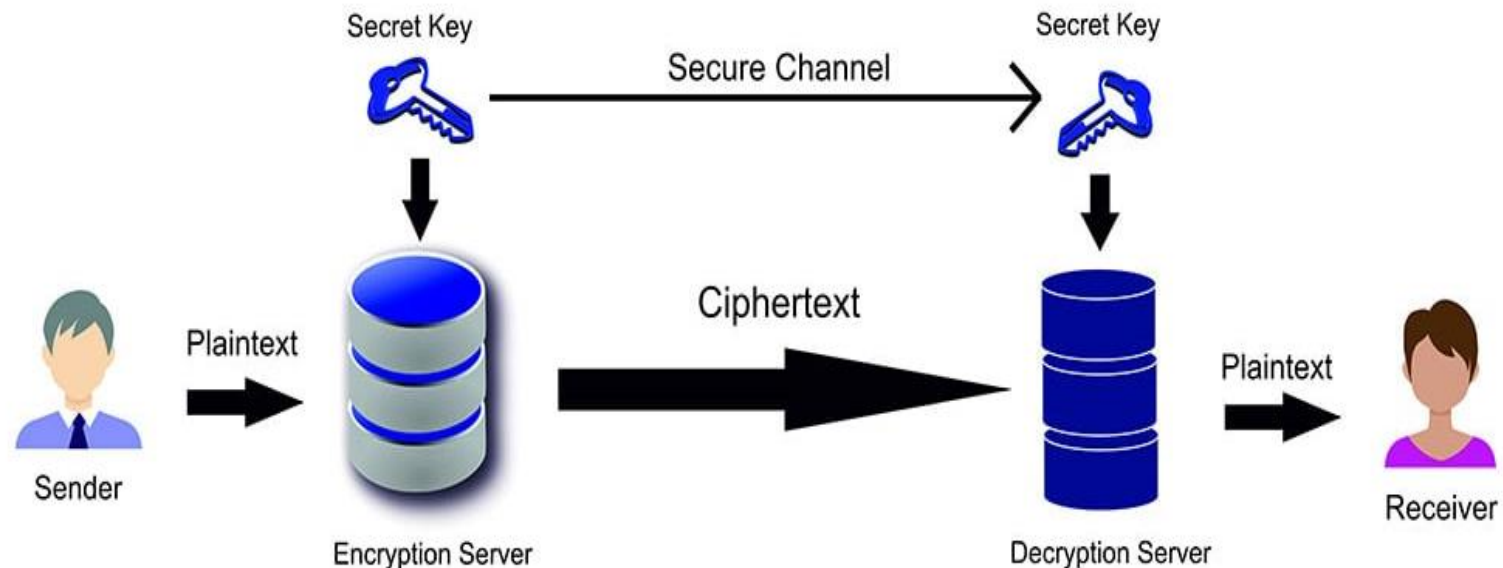
A one-way mathematical transformation of data into a unique fingerprint (MD5, SHA-256, SHA-3).



Internet and computer security

Symmetric Cryptography

Symmetric cryptography is the traditional method of securing data transmitted between two participants in communication. This method of data protection is almost as old as human communication itself. The essence of symmetric cryptography is that for secure communication, both participants must have the same key, which allows them to encrypt and decrypt data



Internet and computer security

Symmetric Cryptography

Key characteristics of symmetric cryptography:

- **One key** - The same key is used for both encryption and decryption operations.
- **Fast and efficient** - Particularly suitable for encrypting large amounts of data.
- **Security challenge** - The key must be securely shared between the sender and the receiver.

The most well-known symmetric algorithms are:

- **AES (Advanced Encryption Standard)**
- **DES (Data Encryption Standard)**
- **3DES (Triple DES)**
- **Blowfish, Twofish, RC4**



Internet and computer security

Example: Mathematical Model of Symmetric Cryptography

Assuming there is a message (plaintext) P , a key K , and an encryption function E , the mathematical model of encryption is as follows:

$$C = E_K(P)$$

where C is the ciphertext.

The decryption process uses the reverse process:

$$P = D_K(C)$$

where D is the decryption function that uses the same key K .

Here is an example of the AES algorithm for encryption and decryption using Python.



Internet and computer security

```
from Crypto.Cipher import AES
import base64

# Function to add padding
def pad(text):
    return text + (16 - len(text) % 16) * ' '

# The key must be 16 bytes long
key = b"1234567890123456" # Tačno 16 bajtova

# Text to encrypt
plain_text = "Osjetljivi podaci"

# Create AES object and encrypt the text
cipher = AES.new(key, AES.MODE_ECB)
encrypted_text = cipher.encrypt(pad(plain_text).encode())

# Display encrypted text
print("Encrypted text:", base64.b64encode(encrypted_text).decode())

# Decrypt the text
decrypted_text = cipher.decrypt(encrypted_text).decode().strip()
print("Decrypted text:", decrypted_text)
```



Internet and computer security

Example: Mathematical Model of Symmetric Cryptography

Display after running the program (AES algorithm):

```
C:\Users\AzraKT\Desktop>python aess.py  
Encrypted text: jVvk8kTGrMWJriqnXQUjHw==  
Decrypted text: Sensitive data
```



Internet and computer security

Asymmetric cryptography

Computer networks enable the rapid exchange of data, but in their early days, they were not designed with strong security in mind.

Instead of using a single key for both encryption and decryption, the concept of a key pair was proposed: a public key and a private key.

The **public key (KP)** is available to everyone and is used to encrypt data.

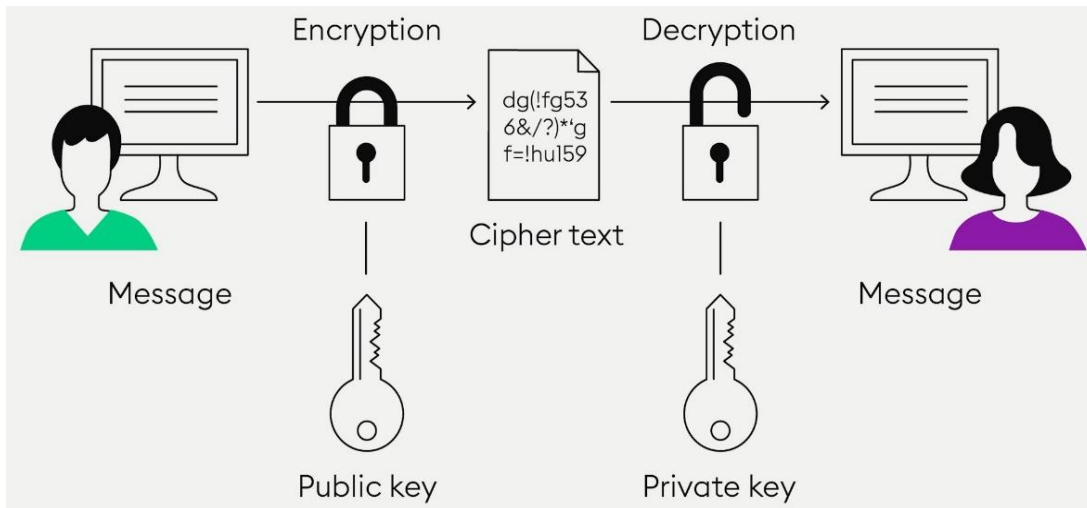
The corresponding **private key (KJ)** is secret and only this key allows for the decryption of data encrypted with the public key from the pair.

Any entity that wants secure communication can publish its public key, and anyone wishing to send an encrypted message uses that key for encryption. Since only the entity holding the private key can decrypt the data, confidentiality is ensured.



Internet and computer security

Asymmetric cryptography



Asymmetric cryptography uses two keys: a **public key** and a **private key**.

It differs from symmetric cryptography, which uses the same key for both encryption and decryption of data.

The public key is used for encrypting data and can be freely shared with others.

The private key is used for decrypting data and must remain private and secure.

Internet and computer security

RSA Algorithm

RSA (*Rivest-Shamir-Adleman*) is one of the most well-known asymmetric cryptographic algorithms. It is based on number theory and is used for encryption and digital signatures.

The public key consists of the numbers e and n , where e is the exponent and n is the product of two large prime numbers.

The private key is the number d , which is related to e and n , and is used for decrypting data.

The RSA algorithm involves three main steps:

1. **Key generation**
2. **Encryption**
3. **Decryption**



Internet and computer security

EXAMPLE 1: RSA Algorithm Implementation in Python

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from base64 import b64encode, b64decode

# 1. Generating RSA keys
def generate_keys():
    key = RSA.generate(2048) # Generate a 2048-bit key
    private_key = key.export_key()
    public_key = key.publickey().export_key()

    # Saving the keys to files
    with open("private.pem", "wb") as f:
        f.write(private_key)

    with open("public.pem", "wb") as f:
        f.write(public_key)

    print("Keys have been successfully generated and saved.")

# 2. Encrypting a message
def encrypt_message(plain_text):
    # Load the public key
    with open("public.pem", "rb") as f:
        public_key = RSA.import_key(f.read())

    # Create a PKCS1_OAEP encryption object
    cipher = PKCS1_OAEP.new(public_key)
```



Internet and computer security

EXAMPLE 1: RSA Algorithm Implementation in Python

```
# Encrypt the message
encrypted_message = cipher.encrypt(plain_text.encode())

# Show the encrypted message in Base64 format
encrypted_message_b64 = b64encode(encrypted_message).decode()
print("Encrypted message:", encrypted_message_b64)

return encrypted_message_b64

# 3. Decrypting the message
def decrypt_message(encrypted_message_b64):
    # Load the private key
    with open("private.pem", "rb") as f:
        private_key = RSA.import_key(f.read())

    # Create a PKCS1_OAEP decryption object
    cipher = PKCS1_OAEP.new(private_key)

    # Convert the Base64 encoded encrypted message back to binary format
    encrypted_message = b64decode(encrypted_message_b64)
```

```
# Decrypt the message
decrypted_message = cipher.decrypt(encrypted_message).decode()
print("Decrypted message:", decrypted_message)

# Main program
def main():
    print("Generating keys...")
    generate_keys() # Generate the keys

    # Encrypting the message
    message = "This is a secret message"
    print("\nEncrypting the message...")
    encrypted_message = encrypt_message(message)

    # Decrypting the message
    print("\nDecrypting the message...")
    decrypt_message(encrypted_message)

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

Internet and computer security

Display after running the program (RSA algorithm):

```
C:\Users\AzraKT\Desktop>python rsaa.py
Generating keys...
Keys have been successfully generated and saved.

Encrypting the message...
Encrypted message: Z/SJI3pFamzOiPOH1iFF4nXwkOg5E9Cx0AC/rLD6RpwmBQp/nr9YSuNCEDTXQwtXSg
7XhUqRVBYpFFKX/kcVC7tn3+clhOUEE7WnGNrrGrZJmtRZaAoZkf1/5uegGZuKFVgAw/N9pTn01/oQtqsVEUa
/M/8Uw9+lMbUXeRIJ5IbDdx9IOh6jIHjFJOCzFfITz6t5jhKR+Nm1iqzt6glmQH3uKJKCLoKdJn242xt/v/aQ
C99vKUiNY84pNXjAJa/Vst2kBlisbH0Jv5AeIsfMthCpeW7A13Apd4eSA16MbBizLZQZJI6iZ7TABMK3htAyJ
EN9XAULmCv5ai9qc8UEcA==

Decrypting the message...
Decrypted message: This is a secret message
```



Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Example of RSA Algorithm Usage (Java):

- Input of the text to be encrypted.
- Generation of public and private keys:
 - The public key is displayed within the program interface.
 - The private key is saved to a file named PK.txt.
- The entered text is encrypted, and the encrypted message is displayed within the interface.
- The program includes decryption of the encrypted message.
- The result of the decryption is shown in the interface.

For the creation of this program, Apache NetBeans IDE and Java JDK were used.

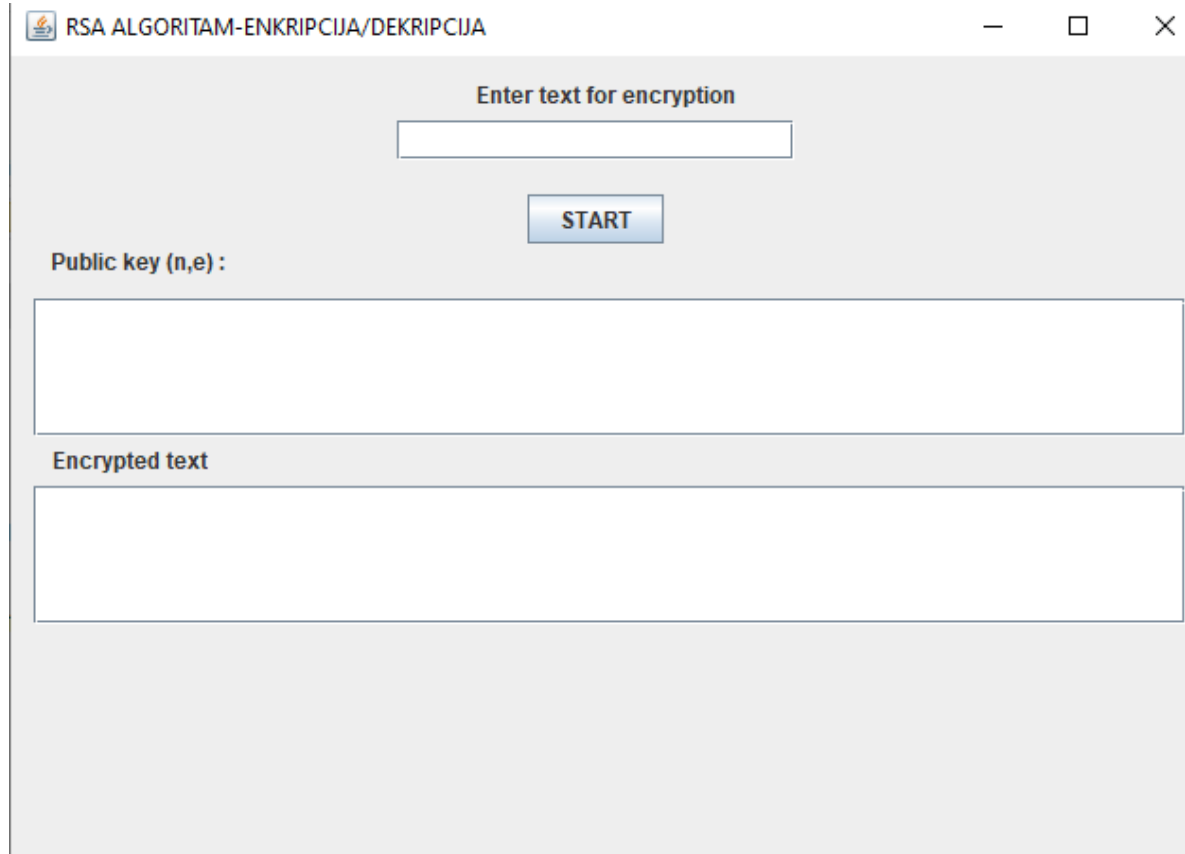
For the UI design, Swing GUI Forms were utilized, and Java was used for implementing the program logic.



Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Interface display of the RSA Algorithm Test program:



The screenshot shows a Java application window titled "RSA ALGORITAM-ENKRIPCIIJA/DEKRIPCIIJA". The interface is light gray and contains the following elements:

- A text input field labeled "Enter text for encryption" at the top.
- A blue "START" button centered below the input field.
- A label "Public key (n,e) :" followed by a large, empty rectangular text area.
- A label "Encrypted text" followed by another large, empty rectangular text area at the bottom.



Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Display of the RSA Algorithm Test function:

```
private void button1ActionPerformed(java.awt.event.ActionEvent evt)
    RSA rsa = new RSA(1024);
    String tekst = this.unosTeksta.getText();
    /* Displaying the public key */
    this.jTextArea1.setText(rsa.getN() + "\n" + rsa.getE());
    BigInteger tekstUBroj = new BigInteger(tekst.getBytes());
    /* Encryption of the entered text */
    BigInteger sifriraniTekst = rsa.enkrijcija(tekstUBroj);
    this.jTextArea2.setText(sifriraniTekst.toString());

    /* Decryption of the encrypted text to the original */
    tekstUBroj = rsa.dekrijcija(sifriraniTekst);
    String text2 = new String(tekstUBroj.toByteArray());
    this.ispisDesifriranogTeksta.setText("Decrypted text: " + text2);
}
```



Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Display of the RSA Algorithm Test function:

```
import java.math.BigInteger;
import java.security.SecureRandom;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class RSA {

    private int bitlen = 1024;
    BigInteger n, d, e;

    /**
     * Creating a class instance for encryption and decryption.
     */
    public RSA(int bits) {
        bitlen = bits;
        this.generirajKljuceve();
    }

    /**
     * Generate a new set of encryption keys
     */
    private synchronized void generirajKljuceve() {
        SecureRandom r = new SecureRandom();
        BigInteger p = new BigInteger(bitlen / 2, 100, r);
```

```
        BigInteger q = new BigInteger(bitlen / 2, 100, r);
        n = p.multiply(q);
        BigInteger m = (p.subtract(BigInteger.ONE)).multiply(q
            .subtract(BigInteger.ONE));
        e = new BigInteger("17");

        /**
         * Use a fixed public key (e.g., e=17) for encryption
         */
        while (m.gcd(e).intValue() > 1) {
            e = e.add(new BigInteger("2"));
        }
        d = e.modInverse(m);

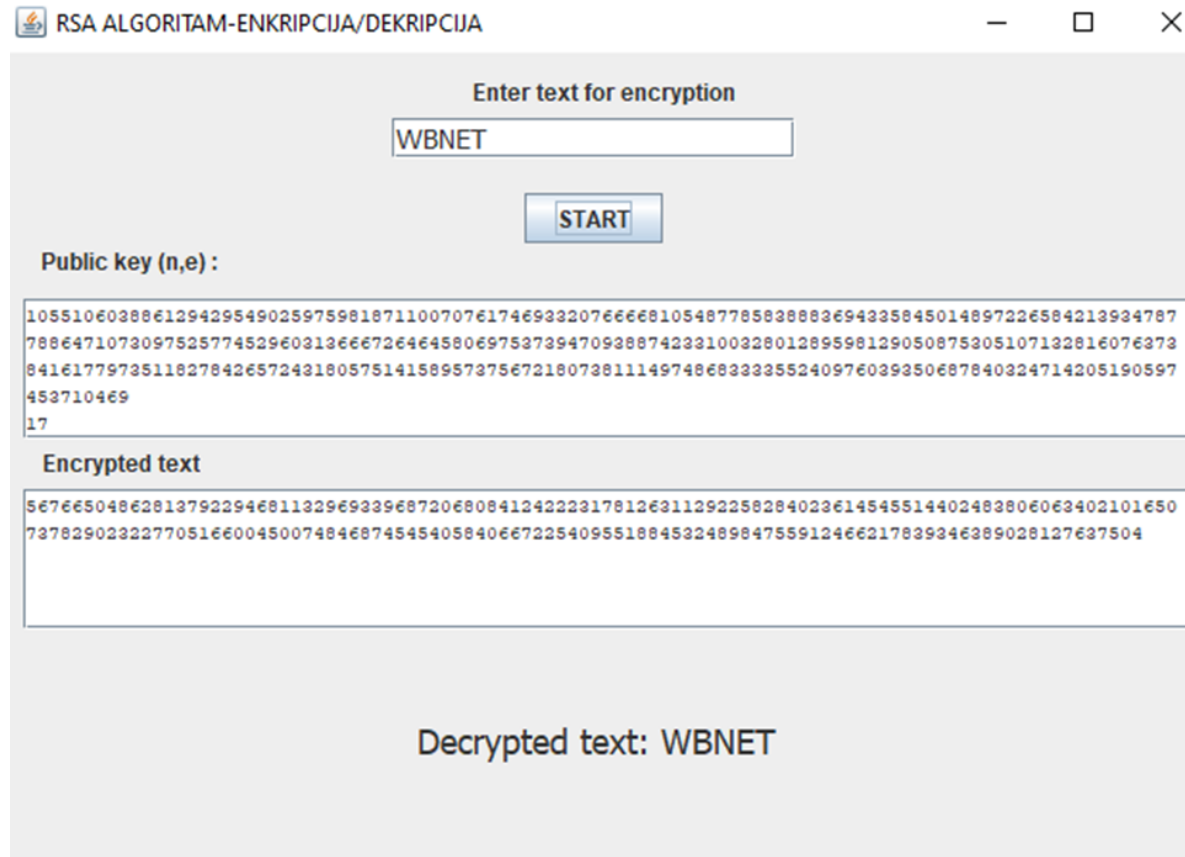
        /**
         * saving the private key to the file PK.txt (p,q,d)
         */
        pisanjePKuFajl(p.toString() + "\n" + q.toString() + "\n" + d.toString());
    }

    private static void pisanjePKuFajl(String data) {
        try {
```

Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Running and testing the application (inputting text):



RSA ALGORITAM-ENKRIPCIJA/DEKRIPCIJA

Enter text for encryption

WBNET

START

Public key (n,e):

1055106038861294295490259759818711007076174693320766668105487785838883694335845014897226584213934787
7886471073097525774529603136667264645806975373947093887423310032801289598129050875305107132816076373
8416177973511827842657243180575141589573756721807381114974868333355240976039350687840324714205190597
453710469
17

Encrypted text

5676650486281379229468113296933968720680841242223178126311292258284023614545514402483806063402101650
7378290232277051660045007484687454540584066722540955188453248984755912466217839346389028127637504

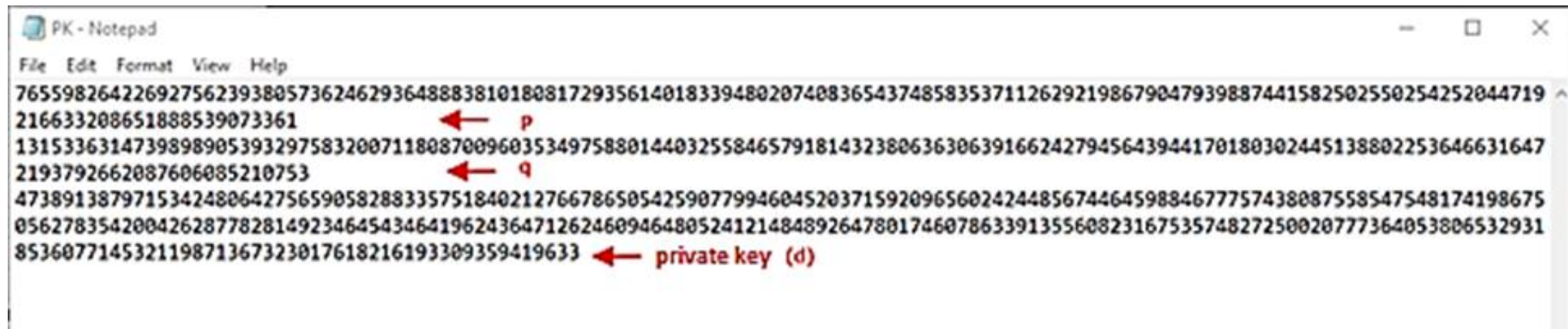
Decrypted text: WBNET



Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Saving the private key in the file PK.txt:



```
PK - Notepad
File Edit Format View Help
7655982642269275623938057362462936488838101808172935614018339480207408365437485835371126292198679047939887441582502550254252044719
216633208651888539073361 ← p
1315336314739898905393297583200711808700960353497588014403255846579181432380636306391662427945643944170180302445138802253646631647
2193792662087606085210753 ← q
4738913879715342480642756590582883357518402127667865054259077994604520371592096560242448567446459884677757438087558547548174198675
0562783542004262877828149234645434641962436471262460946480524121484892647801746078633913556082316753574827250020777364053806532931
853607714532119871367323017618216193309359419633 ← private key (d)
```

Internet and computer security

EXAMPLE 2: RSA Algorithm Implementation in a Java Application

Running and testing the application (entering a number):

RSA ALGORITAM-ENKRIPCJA/DEKRIPCJA

Enter text for encryption

2025

START

Public key (n,e) :

```
1450042658123362037976083163654283423966825852800129278923468792728939147946570468882097555804047294
4410808131187204751420285964340552838620035778492461627116700207019562253781525597119015873067330568
3074394430897867201451864812674679894862621897178712370978637419590383951749197162493727380053059189
382349719
17
```

Encrypted text

```
5376200961525195976688456991687385804477649620627276854016820805771281528539675310926979916030388895
0644016347738872825476888349528748158368748869967861
```

Decrypted text: 2025



Internet and computer security

Hash functions

A **hash function** is a mathematical function that takes input data (or a message) and generates an output value of fixed length, typically called a "hash value" or "hash". Hash functions are used in many security applications, such as data integrity verification, digital signatures, authentication, password storage, etc.

The basic characteristics of hash functions are:

1. One-wayness
2. Fixed output length
3. Uniqueness
4. Sensitivity to changes



Internet and computer security

Hash functions

Popular hash functions include:

- **MD5** (Message Digest Algorithm 5)
 - It is considered outdated due to security vulnerabilities.
- **SHA-1** (Secure Hash Algorithm 1)
 - It is considered insecure due to the possibility of collisions.
- **SHA-256**
 - A part of the SHA-2 family, very secure and widely used.
- **SHA-3**
 - The latest version of the SHA function.



Internet and computer security

EXAMPLE: Implementation of the RSA Algorithm in a Java Application

Implementation and execution of the SHA-256 hash function:

```
import hashlib

# Input text
input_text = "Sensitive data"

# Create SHA-256 hash object
sha256_hash = hashlib.sha256()

# Update the hash object with the input data
sha256_hash.update(input_text.encode())

# Get the hash value
hashed_text = sha256_hash.hexdigest()

# Displaying the original text and its SHA-256 hash
print("Original text:", input_text)
print("SHA-256 Hash value:", hashed_text)
```



```
C:\Users\AzraKT\Desktop>python hash.py
Original text: Sensitive data
SHA-256 Hash value: 474de7276ecc120b83bc42291a96a36c648a0f6428bf7c88bfdccf32bfbb4339
```



Internet and computer security

RNG Generators

Random Number Generators (RNGs) are a key component in many aspects of computer system security and cryptography.

They are used to generate numbers that are, by definition, random and unpredictable. RNGs are the foundation for creating secure keys, digital signatures, authentication, and many other operations in modern information technologies. Given the importance of their role, the quality and security of RNGs directly impact the entire system in which they are used.

There are two main types of RNG generators:

1. Pseudo-random number generators (**PRNGs**) and
2. True random number generators (**TRNGs**).



Internet and computer security

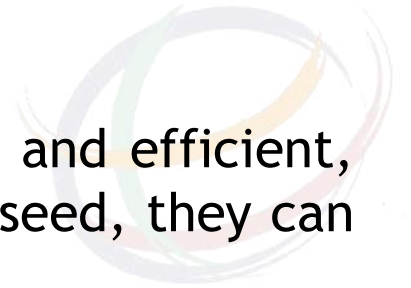
Pseudo-random number generators (PRNG)

Pseudo-random numbers are not "true" random numbers, but are generated using deterministic algorithms.

This means that, although they behave like random numbers, they are actually predictable if the initial condition or "seed" is known.

Pseudo-random number generators are widely used in computer systems due to their speed and efficiency.

One of the most well-known PRNGs is the Mersenne Twister. Although very fast and efficient, PRNGs are not suitable for security applications because if attackers obtain the seed, they can predict the entire sequence of generated numbers.



Internet and computer security

EXAMPLE: PRNG implementation in Python

```
import tkinter as tk
from tkinter import messagebox

# PRNG - Linear Congruential Generator (LCG)
class PRNG:
    def __init__(self, seed=0):
        self.a = 1664525 # Multiplier
        self.c = 1013904223 # Increment
        self.m = 2**32 # Modulus
        self.state = seed

    def generate(self):
        self.state = (self.a * self.state + self.c) % self.m
        return self.state

# GUI Application
class PRNGApp:
    def __init__(self, root):
        self.root = root
        self.root.title("PRNG Generator")
        self.root.geometry("400x300")

        self.prng = PRNG() # Create an instance of PRNG
```

```
        # Label to display generated numbers
        self.output_label = tk.Label(root, text="Generated Numbers:", font=("Arial",
12))
        self.output_label.pack(pady=10)

        # Text box to display the numbers
        self.number_display = tk.Text(root, height=10, width=35)
        self.number_display.pack(pady=10)

        # Seed input field
        self.seed_label = tk.Label(root, text="Enter seed (initial value):",
font=("Arial", 10))
        self.seed_label.pack(pady=5)

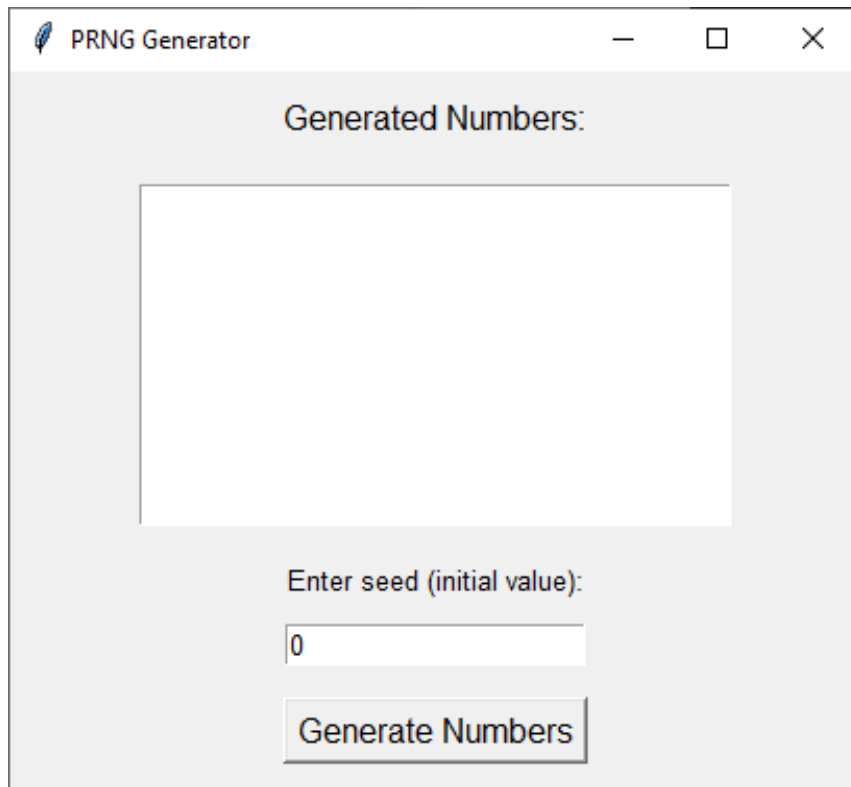
        self.seed_entry = tk.Entry(root, font=("Arial", 10))
        self.seed_entry.pack(pady=5)
        self.seed_entry.insert(0, "0") # Default value for seed

        # Button to generate numbers
        self.generate_button = tk.Button(root, text="Generate Numbers",
font=("Arial", 12), command=self.generate_numbers)
```

Internet and computer security

EXAMPLE: PRNG implementation in Python

Running and testing the application



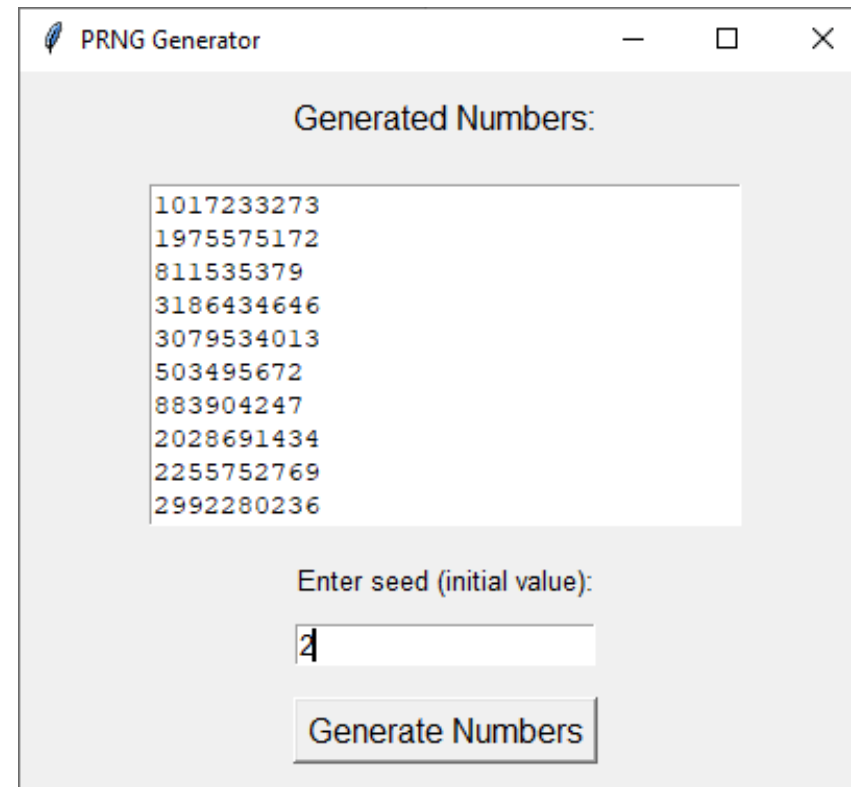
PRNG Generator

Generated Numbers:

Enter seed (initial value):

0

Generate Numbers



PRNG Generator

Generated Numbers:

```
1017233273
1975575172
811535379
3186434646
3079534013
503495672
883904247
2028691434
2255752769
2992280236
```

Enter seed (initial value):

4

Generate Numbers



Internet and computer security

True Random Number Generators (TRNG)

True random number generators (TRNGs) rely on physical processes, such as electronic noise or radioactive decay, to generate unpredictable numbers.

These generators are generally considered more secure because they use an uncertainty source that is non-deterministic and cannot be easily predicted.

However, TRNGs are often slower and more expensive than PRNGs, and they are less commonly used due to the need for specific hardware.



Internet and computer security

EXAMPLE: TRNG Implementation in Python

```
import tkinter as tk
from tkinter import messagebox
import os

# TRNG (True Random Number Generator) using os.urandom()
class TRNG:
    def generate(self):
        # Using os.urandom() which uses entropy from the operating system to generate
        random numbers
        random_bytes = os.urandom(4) # Generate 4 bytes of random data
        random_number = int.from_bytes(random_bytes, "big") # Convert bytes into an
        integer
        return random_number

# GUI Application
class TRNGApp:
    def __init__(self, root):
        self.root = root
        self.root.title("TRNG Generator")
        self.root.geometry("400x300")

        self.trng = TRNG() # Create an instance of TRNG
```

```
12))
        self.output_label.pack(pady=10)

        # Text box to display the numbers
        self.number_display = tk.Text(root, height=10, width=35)
        self.number_display.pack(pady=10)

        # Button to generate numbers
        self.generate_button = tk.Button(root, text="Generate Numbers",
        font=("Arial", 12), command=self.generate_numbers)
        self.generate_button.pack(pady=10)

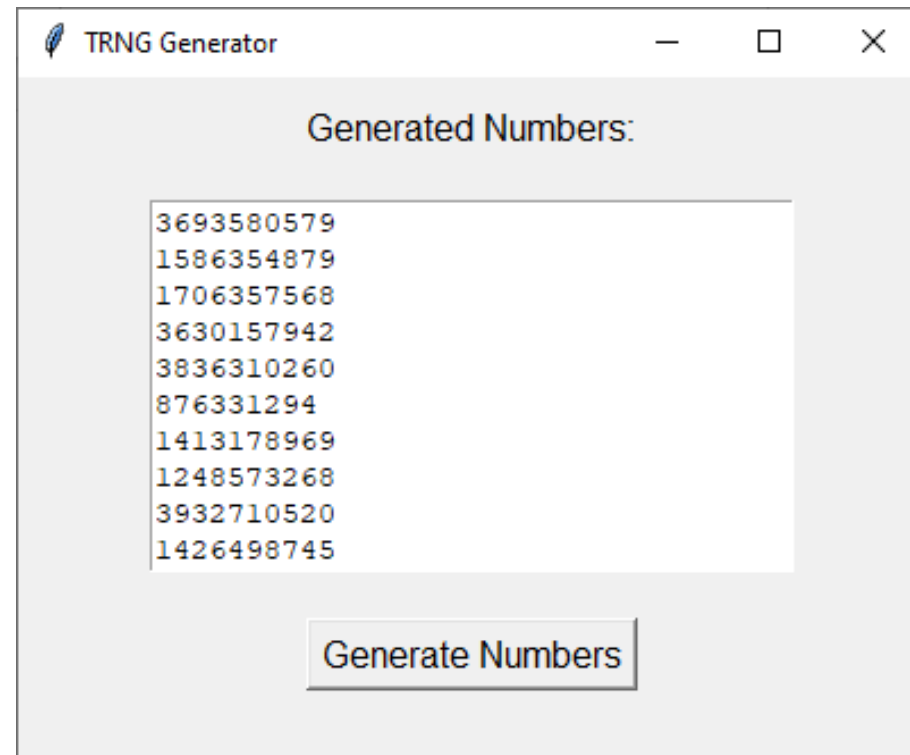
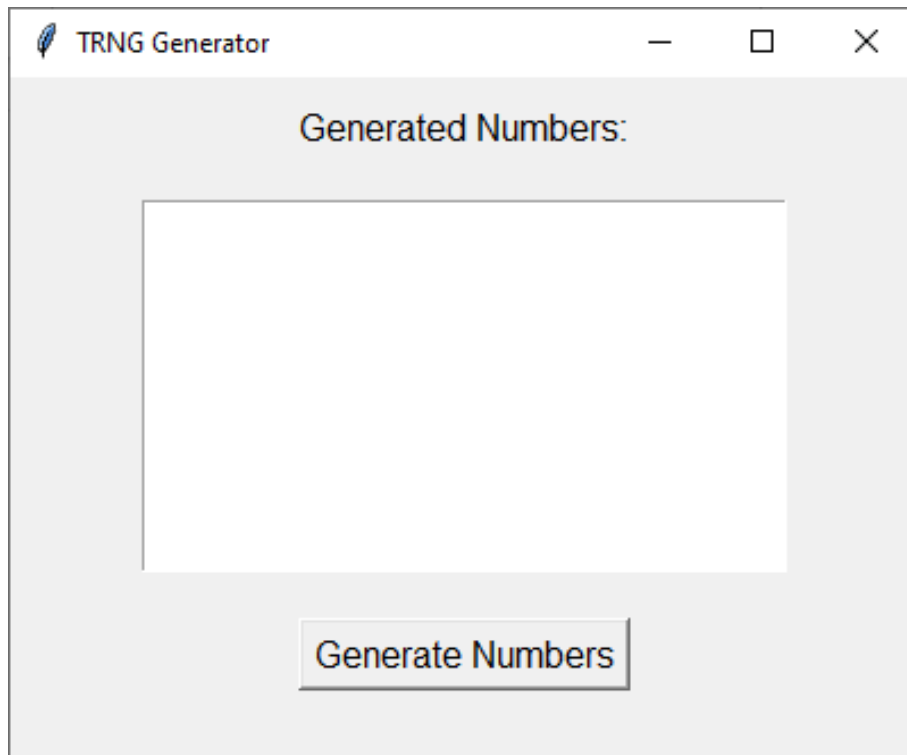
    def generate_numbers(self):
        self.number_display.delete(1.0, tk.END) # Clear previous numbers
        for _ in range(10): # Generate 10 numbers
            random_number = self.trng.generate()
            self.number_display.insert(tk.END, f"{random_number}\n")

# Create the main Tkinter window for the application
if __name__ == "__main__":
```

Internet and computer security

EXAMPLE: TRNG Implementation in Python

Running and testing the application



Internet and computer security

Application of RNG Generators in Cryptography

RNG generators play a crucial role in cryptography, where the security of systems heavily relies on generating random numbers that are difficult to predict.

The main applications of RNGs in cryptography include:

- Generation of cryptographic keys
- Salt and IV (Initialization Vector)
- Digital Signatures and Certificates
- Generation of Session Keys



Internet and computer security

CONCLUSION

By completing this course, participants will gain comprehensive knowledge in the field of internet and computer security, including:

- protecting users and computers on the internet,
- identifying and preventing security threats, and
- implementing protection through cryptographic techniques.

Through practical examples in Python, participants will be enabled to apply their acquired theoretical knowledge in real-world situations, thus developing the necessary skills to implement security solutions in the digital environment. Mastering cryptographic algorithms and data protection methods contributes to their ability to recognize and address security challenges, preparing them to face the ongoing threats in today's dynamic internet landscape.



Co-funded by
the European Union

Questions & Answers

"Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them."

Network of centers for regional short study programs in the countries of the Western Balkans

Call: ERASMUS-EDU-2023-CBHE

Project number: 101128813



UNIVERSITY OF LJUBLJANA
Faculty of Electrical Engineering



University of Pristina
Kosovska Mitrovica

