

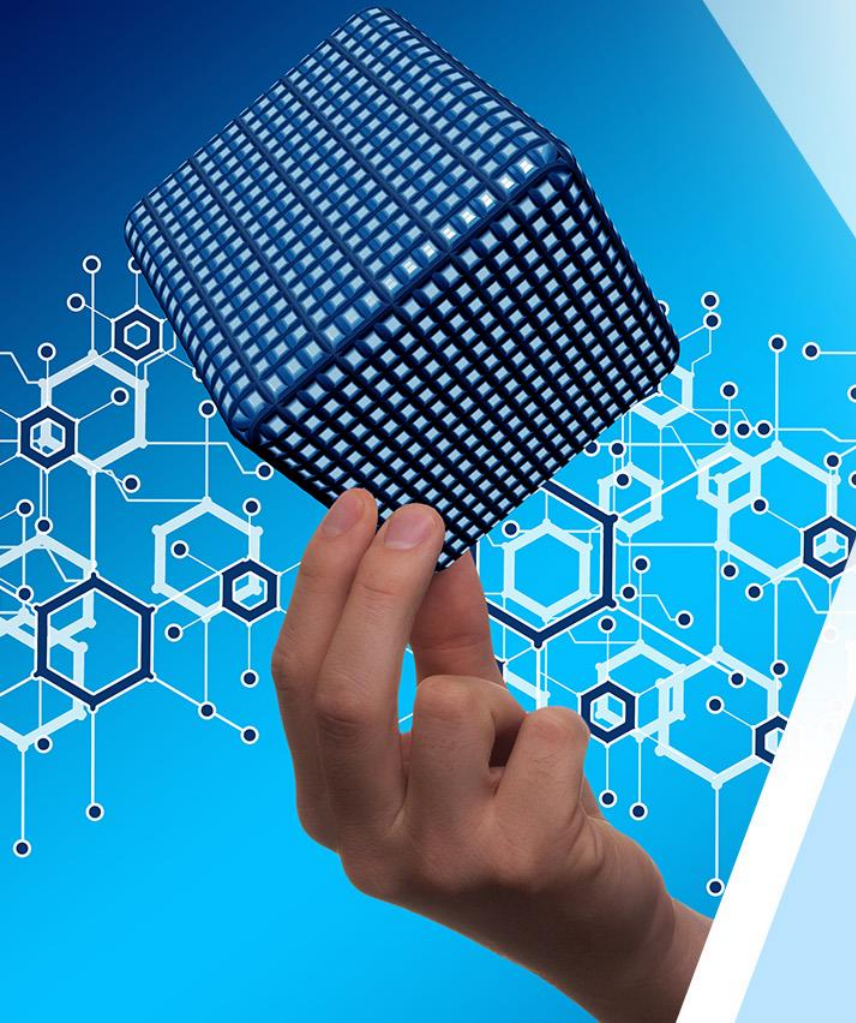
Paralelni računarski sistemi

# APLIKACIJA ZA DOPISIVANJE SA ENKRIPCIJOM I DEKRIPCIJOM

Studenti:

Đorđe Odović, 1740

Milenko Marjanović, 1345





# Uvod

- Paralelizacija i njene primjene na enkripciju i dekripciju
- Korištene tehnologije (C#, .NET, Socket)
- AES enkripcija
- Diffie-Hellman algoritam za razmjenu ključa
- Analiza efikasnosti



# Sadržaj

**Paralelizacija**

**Radno okruženje**

**Tehnologije**

**Enkripcija i dekripcija**

**Izgled i rad aplikacije**

**Testiranje performansi**

**Analiza performansi**

**Zaključak**

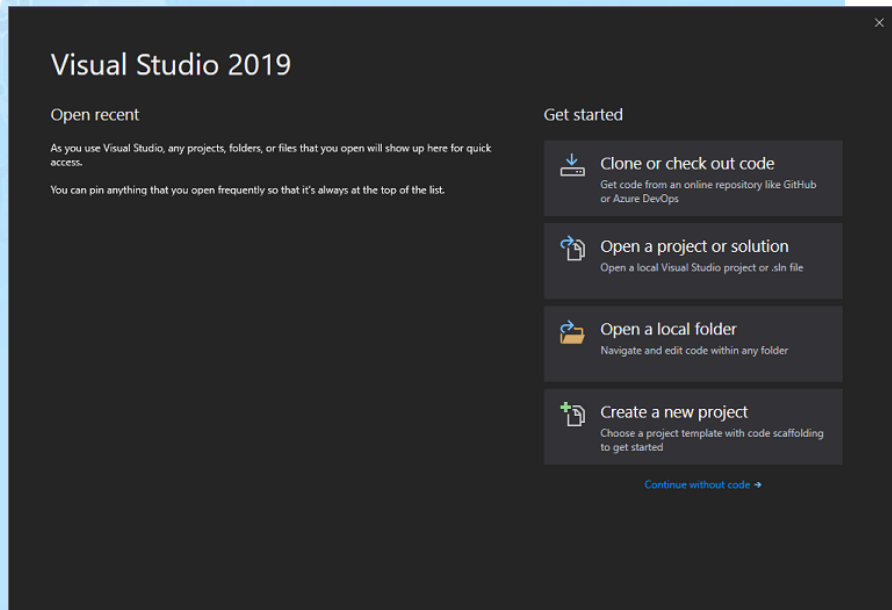


# Paralelizacija

- Sposobnost obavljanja više zadataka istovremeno, a ne uzastopno
- Značajno povećanje brzine i efikasnosti računara

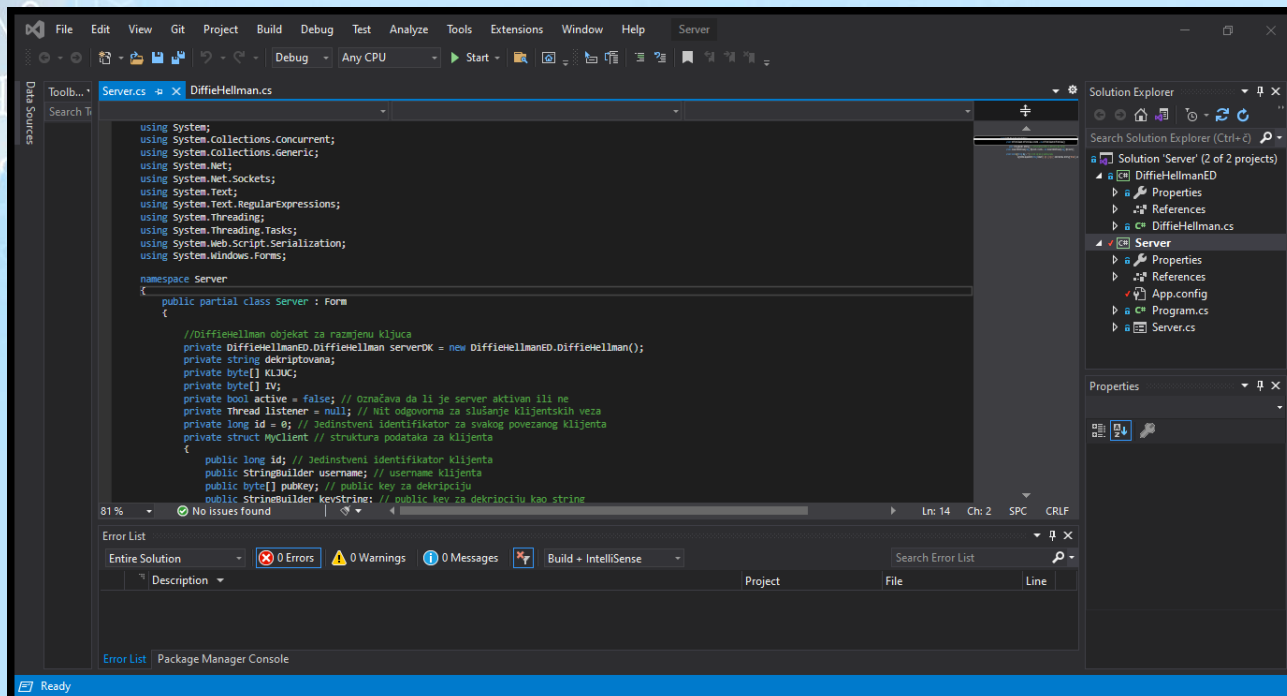
# Radno okruženje

## Visual Studio



# Radno okruženje

## Visual Studio





# Radno okruženje

Windows 10



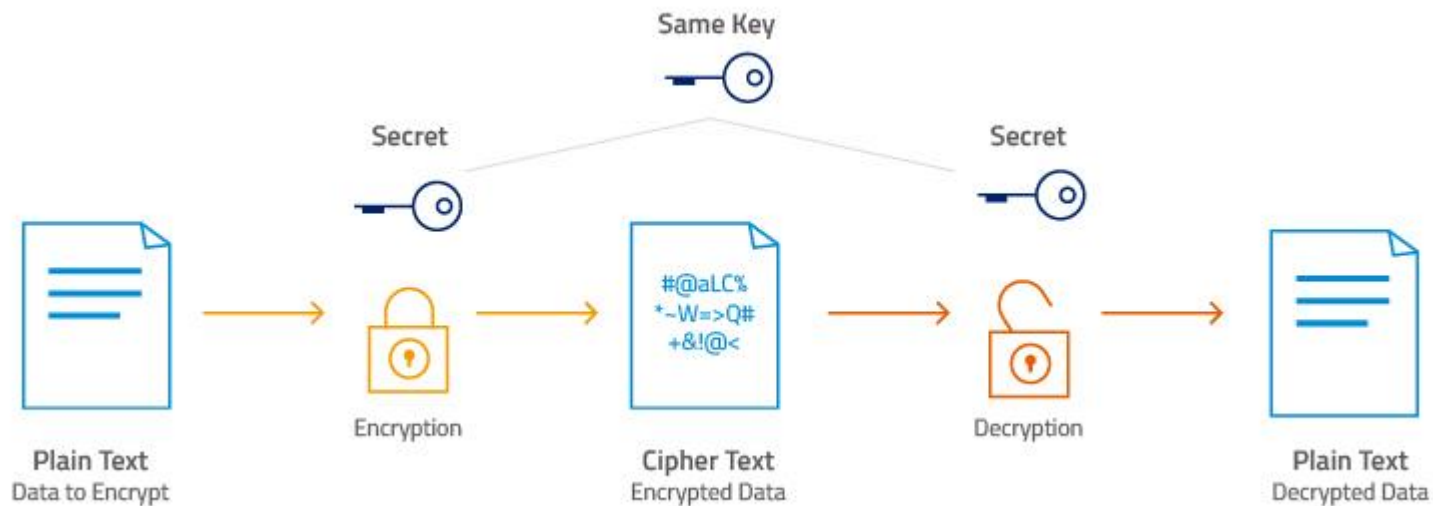
# Tehnologije

- C#
- .NET



# Enkripcija i dekripcija

## AES algoritam





# Enkripcija i dekripcija

## AES algoritam

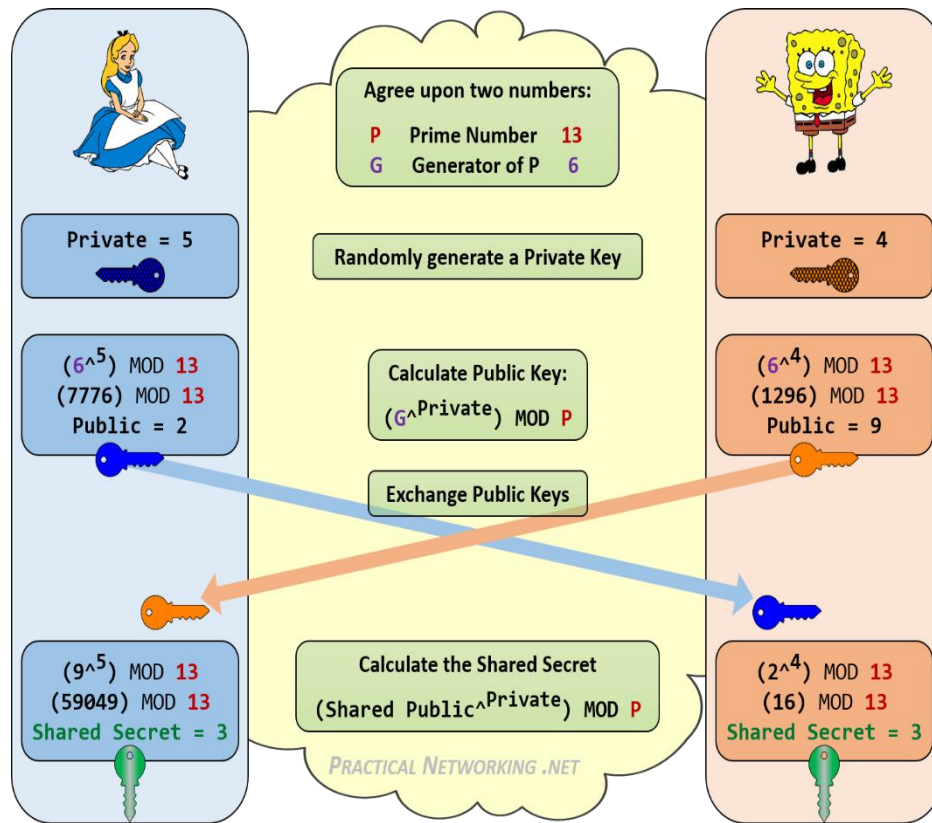
### Veličina ključa

### Moguće kombinacije

1 bit	2
2 bits	4
4 bits	16
8 bits	256
16 bits	65536
32 bits	$4.2 \times 10^9$
56 bits (DES)	$7.2 \times 10^{16}$
64 bits	$1.8 \times 10^{19}$
128 bits (AES)	$3.4 \times 10^{38}$
192 bits (AES)	$6.2 \times 10^{57}$
256 bits (AES)	$1.1 \times 10^{77}$

# Enkripcija i dekripcija

## Diffie-Hellman algoritam



# Izgled i rad aplikacije

## Server

The screenshot shows a window titled "Server" with standard Windows window controls (minimize, maximize, close). The interface is divided into several sections:

- Top Left:** A "Start" button.
- Top Center:** Configuration fields for "Address:" (containing "127.0.0.1") and "Port:" (containing "9000").
- Top Right:** A "Key:" field and a checkbox labeled "Hide key".
- Middle Left:** A "Clear" button.
- Middle Center:** A "Log" label above a large, empty text area with a vertical scrollbar.
- Middle Right:** A "Disconnect all" button.
- Bottom Left:** A "Send" label above a small input field.
- Bottom Right:** A version number "v1.5".
- Right Panel:** A table with three columns: "ID", "Name", and "Disconnect". The table is currently empty.

# Izgled i rad aplikacije

## Server

The screenshot displays the 'Server' application window. It features a configuration section at the top with fields for Address (192.168.183.160), Port (19000), and Username (Server). Below these are buttons for 'Stop', 'Clear', and 'Log', along with a 'Disconnect all' button. A log window shows a series of system messages including client connections and disconnections. On the right, a table lists 11 clients with their IDs, names, and a 'Kick' button. The status bar at the bottom indicates 'Send' and 'v1.5'.

Stop

Address: 192.168.183.160 Port: 19000

Username: Server Key:

☐ Hide key

Total clients: 10

Clear Log Disconnect all

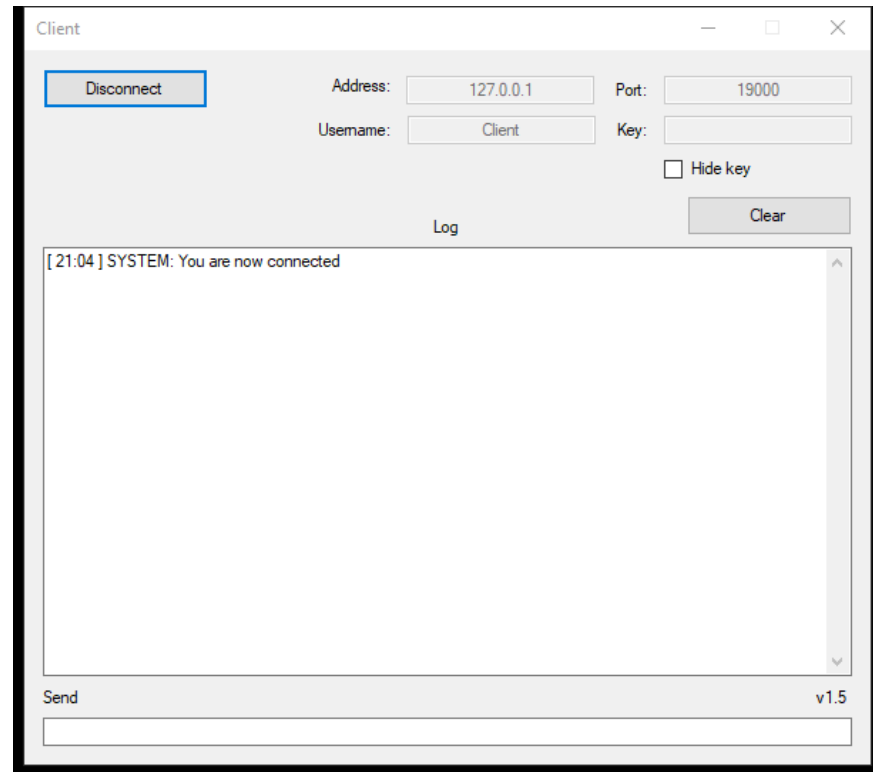
[ 12:28 ] SYSTEM: Server has started  
[ 12:28 ] SYSTEM: Client lokalni has connected  
[ 12:28 ] SYSTEM: ClientUdaljeni has connected  
[ 12:28 ] Client lokalni: sta ima  
[ 12:28 ] ClientUdaljeni: ja sam udaljen  
[ 12:29 ] SYSTEM: Client lok 2 has connected  
[ 12:30 ] SYSTEM: ClientUd2 has connected  
[ 12:30 ] ClientUd2: udaljeni broj 2  
[ 12:31 ] SYSTEM: Clientlok3 has connected  
[ 12:31 ] SYSTEM: ClientUd3 has connected  
[ 12:31 ] Clientlok3: lokalni 3  
[ 12:31 ] ClientUd3: ud 3  
[ 12:31 ] SYSTEM: Clientlok4 has connected  
[ 12:31 ] SYSTEM: ClientUd4 has connected  
[ 12:31 ] Clientlok4: lok4  
[ 12:31 ] ClientUd4: udaljeni 4  
[ 12:32 ] SYSTEM: Clientlok5 has connected  
[ 12:32 ] SYSTEM: Clientlok5 has disconnected  
[ 12:32 ] SYSTEM: Clientlok5 has connected  
[ 12:32 ] SYSTEM: ClientUd5 has connected  
[ 12:32 ] ClientUd5: Udaljeni5

Send v1.5

ID	Name	Disconnect
1	Client lokalni	Kick
2	ClientUdaljeni	Kick
3	Client lok 2	Kick
4	ClientUd2	Kick
5	Clientlok3	Kick
6	ClientUd3	Kick
7	Clientlok4	Kick
8	ClientUd4	Kick
10	Clientlok5	Kick
11	ClientUd5	Kick

# Izgled i rad aplikacije

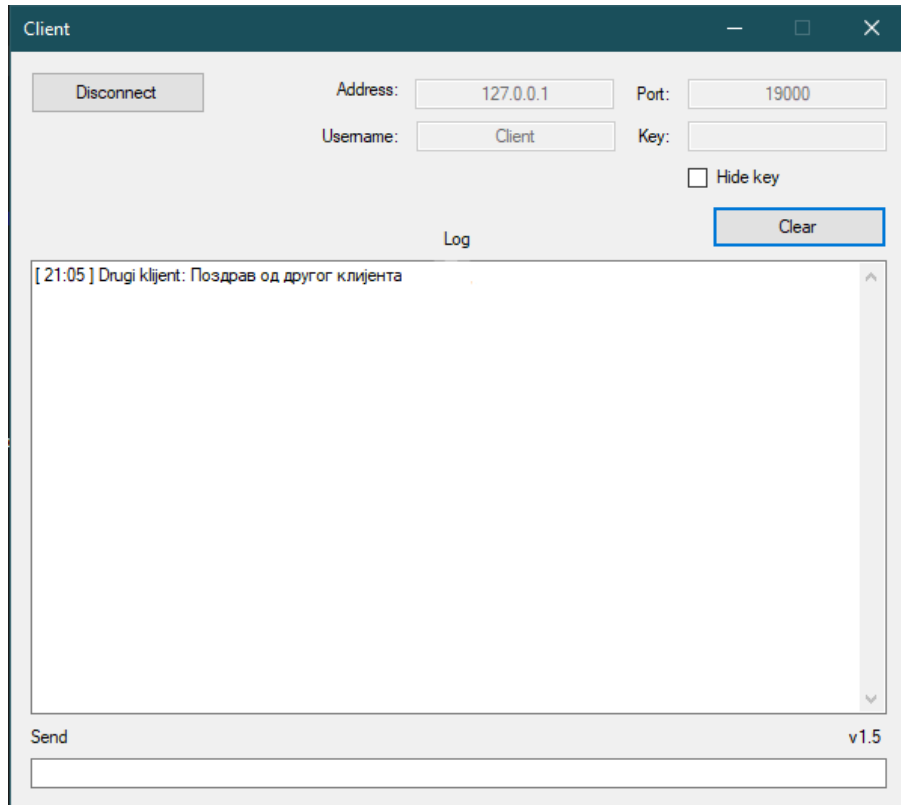
## Klijent





# Izgled i rad aplikacije

Klijent



The screenshot shows a window titled "Client" with standard Windows window controls (minimize, maximize, close). The window contains the following elements:

- A "Disconnect" button in the top left.
- Input fields for "Address" (containing "127.0.0.1") and "Port" (containing "19000").
- Input fields for "Uname:" (containing "Client") and "Key:" (empty).
- A checkbox labeled "Hide key" which is currently unchecked.
- A "Clear" button next to the "Hide key" checkbox.
- A "Log" label above a large text area.
- The text area contains a single log entry: "[ 21:05 ] Drugi klijent: Pozdrav od drugog klijenta".
- A "Send" label and an empty input field at the bottom left.
- The version "v1.5" displayed at the bottom right.

# Izgled i rad aplikacije

## Implementacija Diffie-Hellman algoritma

```
public class DiffieHellman : IDisposable
{
    //polja za referenciranje AES klase, ECDiffieHellman klase i public key
    #region Private Fields
    private Aes aes = null;
    private ECDiffieHellman diffieHellman = null;
    private readonly byte[] publicKey;
    #endregion

    //konstruktor klase
    #region Constructor
    public DiffieHellman()
    {
        this.aes = new AesCryptoServiceProvider();
        this.diffieHellman = ECDiffieHellman.Create();

        // Public key koji se salje drugoj strani
        this.publicKey = this.diffieHellman.PublicKey.ToByteArray();
    }
    #endregion

    // geteri za Public Key i IV (Initialization Vector)
    #region Public Properties
    public byte[] PublicKey
    {
        get
        {
            return this.publicKey;
        }
    }

    public byte[] IV
    {
        get
        {
            return this.aes.IV;
        }
    }
    #endregion
}
```



# Izgled i rad aplikacije

## Enkripcija, dekripcija i razmjena ključa

```
//Metoda za enkripciju
//Koristi public key druge strane i enkriptuje poruku
//Na osnovu public key druge strane se generise derivedKey
//kojim se enkriptuje poruka
public byte[] Encrypt(byte[] publicKey, string secretMessage)
{
    byte[] encryptedMessage;
    using (ECDiffieHellman otherParty = ECDiffieHellman.Create())
    {
        var otherKey = ECDiffieHellmanCngPublicKey.FromByteArray(publicKey, CngKeyBlobFormat.EccPublicBlob);
        byte[] derivedKey = diffieHellman.DeriveKeyMaterial(otherKey);
        aes.Key = derivedKey;

        using (MemoryStream cipherText = new MemoryStream())
        {
            using (CryptoStream cryptoStream = new CryptoStream(cipherText, aes.CreateEncryptor(), CryptoStreamMode.Write))
            {
                byte[] ciphertextMessage = Encoding.UTF8.GetBytes(secretMessage);
                cryptoStream.Write(ciphertextMessage, 0, ciphertextMessage.Length);
            }
            encryptedMessage = cipherText.ToArray();
        }
    }

    return encryptedMessage;
}
```



# Izgled i rad aplikacije

## Enkripcija,dekripcija i razmjena ključa

```
//Metoda za dekripciju
//Koristi public key druge strane i IV
//Na osnovu public key druge strane se generise izvedeni kljuc derivedKey
//na osnovu izvedenog kljuka i iv vrsimo dekripciju
public string Decrypt(byte[] publicKey, byte[] encryptedMessage, byte[] iv)
{
    string decryptedMessage;
    using (ECDiffieHellman otherParty = ECDiffieHellman.Create())
    {
        var otherKey = ECDiffieHellmanCngPublicKey.FromByteArray(publicKey, CngKeyBlobFormat.EccPublicBlob);
        byte[] derivedKey = diffieHellman.DeriveKeyMaterial(otherKey);
        aes.Key = derivedKey;
        aes.IV = iv;
        aes.Padding = PaddingMode.Zeros;

        using (MemoryStream plainText = new MemoryStream())
        {
            using (CryptoStream cryptoStream = new CryptoStream(plainText, aes.CreateDecryptor(), CryptoStreamMode.Write))
            {
                cryptoStream.Write(encryptedMessage, 0, encryptedMessage.Length);
            }
            decryptedMessage = Encoding.UTF8.GetString(plainText.ToArray());
        }
    }

    return decryptedMessage;
}
```



# Izgled i rad aplikacije

## Dekripcija na serveru

```
// ako su pročitani bajtovi
if (bytes > 0)
{
    // Dodati primljene podatke podacima StringBuilder u objektu MyClient
    obj.data.AppendFormat("{0}", Encoding.UTF8.GetString(obj.buffer, 0, bytes));
    try
    {
        // Provjeri da li ima još podataka za čitanje
        if (obj.stream.DataAvailable)
        {
            // Započinjanje druge asinhronne operacije čitanja
            obj.stream.BeginRead(obj.buffer, 0, obj.buffer.Length, new AsyncCallback(Read), obj);
        }
        else
        {
            // Svi podaci su primljeni, obradite poruku
            string msg = string.Format("{0}: {1}", obj.username, obj.data);
            string enkriptovanap = string.Format("{0}", obj.data);
            // Dekripcija primljene poruke

            JavaScriptSerializer json = new JavaScriptSerializer();
            //rječnik za primljene podatke koji se sastoji od ključa i podataka
            Dictionary<string, string> data = json.Deserialize<Dictionary<string, string>>(obj.data.ToString());

            //dekripcija poruke
            if (data.ContainsKey("iv"))
            {
                dekriptovana = serverDK.Decrypt(Convert.FromBase64String(data["publicKey"]), Convert.FromBase64String(data["message"]), Convert.FromBase64String(data["IV"]));
            }
        }
    }
}
```



# Izgled i rad aplikacije

## Sekvencijalna enkripcija na serveru

```
var foreachWatch = System.Diagnostics.Stopwatch.StartNew();
//enkripcija kroz foreach
foreach (KeyValuePair<long, MyClient> klijent in clients)
{
    if (klijent.Value.id != obj.id)
    {
        MyClient tmp = new MyClient();

        long id = klijent.Value.id;
        byte[] pubKey = Convert.FromBase64String(klijent.Value.keyString.ToString());
        byte[] IV = Convert.FromBase64String(klijent.Value.IVString.ToString());
        byte[] secretMessage = serverDK.Encrypt(pubKey, dekriptovanaSaUserInfo);
        tmp.id = id;
        tmp.pubKey = pubKey;
        tmp.buffer = klijent.Value.buffer;
        tmp.IV = IV;
        tmp.handle = klijent.Value.handle;
        tmp.client = klijent.Value.client;
        tmp.data = klijent.Value.data;
        tmp.IVString = klijent.Value.IVString;
        tmp.keyString = klijent.Value.keyString;
        tmp.stream = klijent.Value.stream;
        Send(Poruka(secretMessage), tmp);
        // Briše podatke StringBuilder-a za sledeću poruku
        tmp.data.Clear();
        // Signalizira niti na čekanju da je operacija završena
        tmp.handle.Set();
    }
}

foreachWatch.Stop();
```





# Izgled i rad aplikacije

## Paralelna enkripcija na serveru

```
var parallelWatch = System.Diagnostics.Stopwatch.StartNew();
//enkripcija kroz PARALELNI foreach
Parallel.ForEach(clients, klient =>
{
    if (klijent.Value.id != obj.id)
    {
        MyClient tmp = new MyClient();
        long id = klijent.Value.id;
        byte[] pubKey = Convert.FromBase64String(klijent.Value.keyString.ToString());
        byte[] IV = Convert.FromBase64String(klijent.Value.IVString.ToString());
        byte[] secretMessage = serverDK.Encrypt(pubKey, dekriptovanaSaUserInfo);
        tmp.id = id;
        tmp.pubKey = pubKey;
        tmp.buffer = klijent.Value.buffer;
        tmp.IV = IV;
        tmp.handle = klijent.Value.handle;
        tmp.client = klijent.Value.client;
        tmp.data = klijent.Value.data;
        tmp.IVString = klijent.Value.IVString;
        tmp.keyString = klijent.Value.keyString;
        tmp.stream = klijent.Value.stream;
        Send(Poruka(secretMessage), tmp);
        // Briše podatke StringBuilder-a za sledeću poruku
        tmp.data.Clear();
        // Signalizira niti na čekanju da je operacija završena
        tmp.handle.Set();
    }
});
parallelWatch.Stop();
```

# Testiranje performansi

Broj klijenata	Način izvršavanja	Vrijeme izvršavanja (s)	Razlika u vremenu izvršavanja (paralelno-sekvencijalno) (s)
2.	Sekvencijalno	0.0044485	0.0016243
	Paralelno	0.0060728	
3.	Sekvencijalno	0.0094060	0.0013925
	Paralelno	0.0107985	
4.	Sekvencijalno	0.0210384	0.0014353
	Paralelno	0.0224737	
5.	Sekvencijalno	0.0333167	-0.0071615
	Paralelno	0.0260055	
6.	Sekvencijalno	0.0577922	-0.0266108
	Paralelno	0.0311814	
7.	Sekvencijalno	0.0479412	-0.0041718
	Paralelno	0.0437694	
8.	Sekvencijalno	0.0535184	-0.0342891
	Paralelno	0.0192293	
9.	Sekvencijalno	0.0696097	-0.0100423
	Paralelno	0.0595674	
10.	Sekvencijalno	0.0929398	-0.0322579
	Paralelno	0.0606819	

Testovi na lokalnoj mašini

# Testiranje performansi

Broj klijenata	Način izvršavanja	Vrijeme izvršavanja (s)	Razlika u vremenu izvršavanja (paralelno-sekvencijalno) (s)
2.	Sekvencijalno	0.0105185	0.1105885
	Paralelno	0.1211070	
3.	Sekvencijalno	0.0152706	-0.0034736
	Paralelno	0.0117970	
4.	Sekvencijalno	0.0127824	0.0024558
	Paralelno	0.0152382	
5.	Sekvencijalno	0.0343571	-0.0168304
	Paralelno	0.0175267	
6.	Sekvencijalno	0.0342195	-0.0203842
	Paralelno	0.0138353	
7.	Sekvencijalno	0.0289873	-0.0030966
	Paralelno	0.0258907	
8.	Sekvencijalno	0.0672018	-0.0435892
	Paralelno	0.0236126	
9.	Sekvencijalno	0.0567717	-0.0168512
	Paralelno	0.0399205	
10.	Sekvencijalno	0.0647226	-0.0331238
	Paralelno	0.0315988	

Kombinovani testovi

# Testiranje performansi

Broj klijenata	Način izvršavanja	Vrijeme izvršavanja (s)	Razlika u vremenu izvršavanja (paralelno-sekvencijalno) (s)
2.	Sekvencijalno	0.0101870	-0.0005652
	Paralelno	0.0096218	
3.	Sekvencijalno	0.0185927	-0.0057801
	Paralelno	0.0128126	
4.	Sekvencijalno	0.0217809	-0.01221686
	Paralelno	0.0096123	
5.	Sekvencijalno	0.0190844	-0.0086972
	Paralelno	0.0103872	
6.	Sekvencijalno	0.0458459	-0.0335805
	Paralelno	0.0122654	
7.	Sekvencijalno	0.0234431	-0.0126663
	Paralelno	0.0107768	
8.	Sekvencijalno	0.0480330	-0.0353858
	Paralelno	0.0126472	
9.	Sekvencijalno	0.0592942	-0.0425470
	Paralelno	0.0167472	
10.	Sekvencijalno	0.0547889	-0.0356627
	Paralelno	0.0191262	

Testovi sa udaljenim klijentima



# Analiza performansi

- Izraženiji uticaj paralelizma sa povećanjem broja klijenata
- Povećano vrijeme izvršavanja sa malim brojem klijenata
- Očit uticaj pozadinskih procesa

# Zaključak

- Efikasnost i isplativost paralelizma zavisi od specifičnog zadatka
- Bitan uticaj pozadinskih procesa
- Krucijalna uloga planera resursa (resource scheduler)





**Hvala na pažnji!**