LAMPIRAN

LAMPIRAN

KODE ENKRIPSI DAN DEKRIPSI

```
private void encryptMenuItem_Click( object sender, EventArgs e )
{
    if( inputTextBox.Text.Length != 0 )
    {
        openFileDialog.FileName = "";
        openFileDialog.Title = "Open Public Key File";
        openFileDialog.Filter = "Public Key Document( *.pke )|*.pke";
        string fileString = null;
        if( openFileDialog.ShowDialog() == DialogResult.OK )
        {
            if( File.Exists( openFileDialog.FileName ) )
            {
                StreamReader streamReader = new StreamReader(
                openFileDialog.FileName, true );
                fileString = streamReader.ReadToEnd();
                streamReader.Close();
                if( fileString.Length >= inputTextBox.MaxLength )
                {
                    MessageBox.Show( "ERROR: \nThe file you are
                    trying to open is too big for the text editor to display
                    properly.\nPlease open a smaller document!\nOperation
                    Aborted!" );
                }
            }
        }
```

```
if( fileString != null )
{
    FinishedProcessDelegate finishedProcessDelegate = new
    FinishedProcessDelegate( FinishedProcess );
    UpdateTextDelegate updateTextDelegate = new
    UpdateTextDelegate( UpdateText );
    string bitStrengthString = fileString.Substring( 0,
    fileString.IndexOf( "</BitStrength>" ) + 14 );
    fileString = fileString.Replace( bitStrengthString, "" );
    int bitStrength = Convert.ToInt32( bitStrengthString.Replace(
    "<BitStrength>", "" ).Replace( "</BitStrength>", "" ) );
    Point point = new Point( ( inputTextBox.Size.Width / 2 ) - (
    panel.Size.Width / 2 ), ( inputTextBox.Size.Height / 2 ) - (
    panel.Size.Height / 2 ) );
    panel.Location = point;
    panel.Visible = true;
    this.Refresh();
    fileMenuItem.Enabled = false;
    editMenuItem.Enabled = false;
    formatMenuItem.Enabled = false;
    encryptionMenuItem.Enabled = false;
    helpMenuItem.Enabled = false;
    if( fileString != null )
    {
        try
        {
            EncryptionThread encryptionThread = new
            EncryptionThread();
```

```
                    Thread encryptThread = new Thread(
                    encryptionThread.Encrypt );
                    encryptThread.IsBackground = true;
                    encryptThread.Start( new Object[] { this,
                    finishedProcessDelegate, updateTextDelegate,
                    inputTextBox.Text, bitStrength, fileString } );
                }
                catch( CryptographicException CEx )
                {
                    MessageBox.Show( "ERROR: \nOne of the following
                    has occured.\nThe cryptographic service provider
                    cannot be acquired.\nThe length of the text being
                    encrypted is greater than the maximum allowed
                    length.\nThe OAEP padding is not supported on this
                    computer.\n" + "Exact error: " + CEx.Message );
                }
                catch( Exception Ex )
                { MessageBox.Show( "ERROR: \n" + Ex.Message ); }
            }
        }
    }
    else
    { MessageBox.Show( "ERROR: You Can Not Encrypt A NULL Value!!!"
    ); }
}
```

```
public void Encrypt( object inputObject )
{
    object[] inputObjects = ( object[] )inputObject;
    containerControl = ( Form ) inputObjects[ 0 ];
    finishedProcessDelegate = ( Delegate ) inputObjects[ 1 ];
    updateTextDelegate = ( Delegate )inputObjects[ 2 ];
    string encryptedString = EncryptString( ( string )inputObjects[ 3 ], ( int
    )inputObjects[ 4 ], ( string )inputObjects[ 5 ] );
    containerControl.Invoke( updateTextDelegate, new object[] {
    encryptedString } );
    containerControl.Invoke( finishedProcessDelegate );
}

public string EncryptString( string inputString, int dwKeySize, string xmlString
)
{
    RSACryptoServiceProvider rsaCryptoServiceProvider = new
    RSACryptoServiceProvider( dwKeySize );
    rsaCryptoServiceProvider.FromXmlString( xmlString );
    int keySize = dwKeySize / 8;
    byte[] bytes = Encoding.UTF32.GetBytes( inputString );
    int maxLength = keySize - 42;
    int dataLength = bytes.Length;
    int iterations = dataLength / maxLength;
    StringBuilder stringBuilder = new StringBuilder();
    for( int i = 0; i <= iterations; i++ )
    {
        byte[] tempBytes = new byte[ ( dataLength - maxLength * i >
        maxLength ) ? maxLength : dataLength - maxLength * i ];
```

```
            Buffer.BlockCopy( bytes, maxLength * i, tempBytes, 0,
            tempBytes.Length );
            byte[] encryptedBytes = rsaCryptoServiceProvider.Encrypt(
            tempBytes, true );
            Array.Reverse( encryptedBytes );
            stringBuilder.Append( Convert.ToBase64String( encryptedBytes ) );

        }
        return stringBuilder.ToString();
}

private void decryptMenuItem_Click( object sender, EventArgs e )
{
        if( inputTextBox.Text.Length != 0 )
        {
            openFileDialog.FileName = "";
            openFileDialog.Title = "Open Private Key File";
            openFileDialog.Filter = "Private Key Document( *.kez )|*.kez";
            string fileString = null;
            if( openFileDialog.ShowDialog() == DialogResult.OK )
            {
                if( File.Exists( openFileDialog.FileName ) )
                {
                    StreamReader streamReader = new StreamReader(
                    openFileDialog.FileName, true );
                    fileString = streamReader.ReadToEnd();
                    streamReader.Close();
                    if( fileString.Length >= inputTextBox.MaxLength )
```

```
            { MessageBox.Show( "ERROR: \nThe file you are trying to
            open is too big for the text editor to display
            properly.\nPlease open a smaller document!\nOperation
            Aborted!" ); }
        }
    }
    if( File.Exists( openFileDialog.FileName ) )
    {
        string bitStrengthString = fileString.Substring( 0,
        fileString.IndexOf( "</BitStrength>" ) + 14 );
        fileString = fileString.Replace( bitStrengthString, "" );
        int bitStrength = Convert.ToInt32( bitStrengthString.Replace(
        "<BitStrength>", "" ).Replace( "</BitStrength>", "" ) );
        Point point = new Point( ( inputTextBox.Size.Width / 2 ) - (
        panel.Size.Width / 2 ), ( inputTextBox.Size.Height / 2 ) - (
        panel.Size.Height / 2 ) );
        panel.Location = point;
        panel.Visible = true;
        this.Refresh();
        fileMenuItem.Enabled = false;
        editMenuItem.Enabled = false;
        formatMenuItem.Enabled = false;
        encryptionMenuItem.Enabled = false;
        helpMenuItem.Enabled = false;
        string tempStorage = inputTextBox.Text;
        if( fileString != null )
        {
            FinishedProcessDelegate finishedProcessDelegate = new
            FinishedProcessDelegate( FinishedProcess );
```

```
            UpdateTextDelegate updateTextDelegate = new
            UpdateTextDelegate( UpdateText );
            try
            {
                EncryptionThread decryptionThread = new
                EncryptionThread();
                Thread decryptThread = new Thread(
                decryptionThread.Decrypt );
                decryptThread.IsBackground = true;
                decryptThread.Start( new Object[] { this,
                finishedProcessDelegate, updateTextDelegate,
                inputTextBox.Text, bitStrength, fileString } );
            }
            catch( CryptographicException CEx )
            {
                MessageBox.Show( "ERROR: \nOne of the following
                has occured.\nThe cryptographic service provider
                cannot be acquired.\nThe length of the text being
                encrypted is greater than the maximum allowed
                length.\nThe OAEP padding is not supported on this
                computer.\n" + "Exact error: " + CEx.Message );
            }
            catch( Exception Ex )
            {
                MessageBox.Show( "ERROR:\n" + Ex.Message );
                SetText( tempStorage );
            }
        }
    }
```

```
        }
        else
        { MessageBox.Show( "ERROR: You Can Not Decrypt A NULL Value!!!"
        ); }
}

public void Decrypt( object inputObject )
{
        object[] inputObjects = ( object[] )inputObject;
        containerControl = ( Form )inputObjects[ 0 ];
        finishedProcessDelegate = ( Delegate )inputObjects[ 1 ];
        updateTextDelegate = ( Delegate )inputObjects[ 2 ];
        string decryptedString = DecryptString( ( string )inputObjects[ 3 ], ( int
        )inputObjects[ 4 ], ( string )inputObjects[ 5 ] );
        containerControl.Invoke( updateTextDelegate, new object[] {
        decryptedString } );
        containerControl.Invoke( finishedProcessDelegate );
}

public string DecryptString( string inputString, int dwKeySize, string xmlString
)
{
        RSACryptoServiceProvider rsaCryptoServiceProvider = new
        RSACryptoServiceProvider( dwKeySize );
        rsaCryptoServiceProvider.FromXmlString( xmlString );
        int base64BlockSize = ( ( dwKeySize / 8 ) % 3 != 0 ) ? ( ( ( dwKeySize / 8 )
        / 3 ) * 4 ) + 4 : ( ( dwKeySize / 8 ) / 3 ) * 4;
        int iterations = inputString.Length / base64BlockSize;
        ArrayList arrayList = new ArrayList();
```

```
        for( int i = 0; i < iterations; i++ )
        {
            byte[] encryptedBytes = Convert.FromBase64String(
            inputString.Substring( base64BlockSize * i, base64BlockSize ) );
            Array.Reverse( encryptedBytes );
            arrayList.AddRange( rsaCryptoServiceProvider.Decrypt(
            encryptedBytes, true ) );
        }
        return Encoding.UTF32.GetString( arrayList.ToArray( Type.GetType(
        "System.Byte" ) ) as byte[] );
}
```

Kode Pembuatan Kunci

```
private void generateKeyPairMenuItem_Click( object sender, EventArgs e )
{
        KeyPairGeneratorForm generator = new KeyPairGeneratorForm();
        if( generator.ShowDialog() == DialogResult.OK )
        {
            RSACryptoServiceProvider RSAProvider = new
            RSACryptoServiceProvider( currentBitStrength );
            string publicAndPrivateKeys = "<BitStrength>" +
            currentBitStrength.ToString() + "</BitStrength>" +
            RSAProvider.ToXmlString( true );
            string justPublicKey = "<BitStrength>" +
            currentBitStrength.ToString() + "</BitStrength>" +
            RSAProvider.ToXmlString( false );
            if( saveFile( "Save Public/Private Keys As", "Public/Private Keys
            Document( *.kez )|*.kez", publicAndPrivateKeys ) )
            {
```

```
                while( !saveFile( "Save Public Key As", "Public Key Document(
                *.pke )|*.pke", justPublicKey ) ) { ; }
            }
        }
}

private void generateKeysButton_Click( object sender, System.EventArgs e )
{
    ProgramRSA.MainForm.SetBitStrength( Convert.ToInt32(
    numericUpDown.Value ) );
    this.DialogResult = DialogResult.OK;
    this.Dispose( true );
}
```

TABEL ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com