How to work with Encrypt and Decrypt Certificate

[Част 1 – Създаване на сертификат/key и криптиране 1](#_Toc36795708)

[1. Как да създадем AES key, това не е сертификат! 1](#_Toc36795709)

[2. Как да криптираме с AES key 1](#_Toc36795710)

[3. Как да създадем Self-signed certificate 2](#_Toc36795711)

[4. Как да криптираме със Self-Signed certificate 2](#_Toc36795712)

[5. Как да криптираме със Windows Data Protection API certificate 3](#_Toc36795713)

[6. Как да създадем certificate issued by CA to encrypt and decrypt 3](#_Toc36795714)

[Част 2 – Използване на сертификата за декриптиране и създаване на обект 3](#_Toc36795715)

[1. Използвайки AES key 3](#_Toc36795716)

[2. Използвайки Self-signed certificate 4](#_Toc36795717)

[3. Използвайки Windows Data Protection API 4](#_Toc36795718)

[4. Използвайки certificate issued by CA 4](#_Toc36795719)

[Част 3 – Как да разчетем SecureString 4](#_Toc36795720)

Част 1 – Създаване на сертификат/key и криптиране

# Как да създадем AES key, това не е сертификат!

Тук, при създаването на ключа, може да се направи с 16 байта Key за криптиране, 24 байта, 32 байта = 256 бита или още както е известно AES 256. Този ключ, може да се използва от всеки, който го има и няма ограничение от един потребител или един компютър при използването.

$KeyFile = "\\Machine1\SharedPath\AES.key"

$Key = **New-Object** Byte[] 16   *# You can use 16, 24, or 32 for AES*

[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($Key)

$Key | **out-file** $KeyFile

# Как да криптираме с AES key

$File = "\\Machine1\SharedPath\Password.txt"

$Password = "P@ssword1" | **ConvertTo-SecureString** -AsPlainText –Force

*<# Or use:*

*$Password = (Get-Credential).Password | ConvertTo-SecureString*

*$Password = Read-Host "Enter Password" -AsSecureString | ConvertTo-SecureString #>*

$Password | **ConvertFrom-SecureString** -key $key | **Out-File** $File

# Как да създадем Self-signed certificate

$svcAccountName = **Read-Host** "Write what will be the service account using this certificate:"

$Months = **Read-Host** "Write how many months do you want the certificate to be valid (only numbers or script will fail):"

**New-SelfSignedCertificate** -KeyDescription "PowerShell Script Encryption-Decryption Key" `

-Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" `

-KeyFriendlyName "PSScriptEncryptDecryptKey" `

-FriendlyName "$svcAccountName-PSScriptCipherCert" `

-Subject "$svcAccountName-PSScriptCipherCert" `

-KeyUsage "DataEncipherment" `

-Type "DocumentEncryptionCert" `

-HashAlgorithm "sha256" `

-CertStoreLocation "Cert:\CurrentUser\My" `

-NotAfter (**Get-Date**).AddMonths($Months)

# Как да криптираме със Self-Signed certificate

Този скрипт взима вече създаден сертификат, username, domain, password, sharedPath. Криптира паролата със сертификата и запазва криптираното във файл, който се намира на посочения sharedPath. След това, записва username@domain във файл (некриптирано), който се намира на посочения sharedPath. Накрая прочита съдържанието от двата файла и го изписва на терминала.

Скрипта е така написан, че да може всеки Service Account да бъде криптиран със свой сертификат и според акаунта който посочваш, когато те попита. Така Import-ва неговия сертификат за да декриптира паролата и всички акаунти да са криптирани с отделен сертификат, а не с 1 общ. Разбира се може всички да се криптират с един.

$svcAccountName = **Read-Host** "Write what will be the service account using this certificate"

$svcDomain = **Read-Host** "Write what will be the domain for this account"

**Write-output** "Requesting the service account password. (This will be encrypted)"

$cred = **Get-Credential** -Message "Enter the service account password that will be used to execute scripts." -UserName ($svcAccountName+‘@’+$svcDomain)

$ImportedCert = **Get-ChildItem** cert:\currentuser\my | **Where-Object** {$\_.Subject -match "$($svcAccountName+’-PSScriptCipherCert’)"}

*# Convert certificate subject to cn= format.*

$certCn =  "cn=$($svcAccountName+’-PSScriptCipherCert’)"

**Write-output** "Encrypting service account password."

$svcAccountUsername = $cred.GetNetworkCredential().Username

$svcAccountPassword = $cred.GetNetworkCredential().Password

*# Encrypt password*

$EncryptedPwd = $svcAccountPassword | **Protect-CmsMessage** -To $certCn

$sharedPath = **Read-Host** "Write where will be stored the password and username files in the format C:\temp\ (not C:\temp ) or \\server-01\shared\ (not \\server-01\shared )”

$upnFilePath = $sharedPath+$svcAccountName+’-upn.txt’

$pwFilePath = $sharedPath+$svcAccountName+'-pw.txt'

*# Write service account username to UPN file*

$svcAccountUpn = $svcAccountName+‘@’+$svcDomain

**Write-output** "Exporting service account username with domain: $svcAccountUpn to $upnFilePath."

**Set-Content** -Path $upnFilePath -Value $svcAccountUpn -Force -Verbose

*# Write encrypted password to shared password file*

**Write-output** "Exporting service account password for $svcAccountName to $pwFilePath."  -Verbose

**Set-Content** -Path $pwFilePath -Value $EncryptedPwd -Force -Verbose

*# Show username*

**Write-output** "Showing exported [username] $svcAccountName from $upnFilePath."  -Verbose

**Write-Output** "Service account name"

**Get-Content** -Path $upnFilePath -Verbose

*# Show encrypted password*

**Write-output** "Showing exported encrypted [password] for $svcAccountName from $pwFilePath."  -Verbose

**Get-Content** -Path $pwFilePath -Verbose

# Как да криптираме със Windows Data Protection API certificate

Ограничение: Този начин, позволява сертификата да се използва от 1 акаунт, на 1 компютър!

Идеята е да имаме SecureString, който да превърнем в обикновен криптиран String. Криптирането по подразбиране, става посредством Windows Data Protection API. От тук идва ограничението, че декриптирането може да се случи само с този акаунт и на този компютър, на който е криптирано.

Пример:

**Read-Host** "Enter Password" -AsSecureString | **ConvertFrom-SecureString** | **Out-File** "C:\Temp\Password.txt"

Друг начин е, да извикаме **Get-Credential**, който е от тип SecureString.

Пример:

(**Get-Credential**).Password | **ConvertFrom-SecureString** | **Out-File** "C:\Temp\Password.txt"

Трети начин е, да вземем обикновен String, да го превърнем в SecureString и от там е същото както по-горе.

Пример:

"P@ssword1" | **ConvertTo-SecureString** -AsPlainText -Force | **ConvertFrom-SecureString** | **Out-File** "C:\Temp\Password.txt"

# Как да създадем certificate issued by CA to encrypt and decrypt

Част 2 – Използване на сертификата за декриптиране и създаване на обект

Обектите, които ще създадем са два. Първият е SecureString, който може и да не използваме. Вторият е CredentialObject, който се използва в скриптовете да предадем Username@Domain и Password.

# Използвайки AES key

Пример – за SecureString:

$PasswordFile = "\\Machine1\SharedPath\Password.txt"

$KeyFile = "\\Machine1\SharedPath\AES.key"

$Key = **Get-Content** $KeyFile

$Password = **Get-Content** $PasswordFile

$Password | **ConvertTo-SecureString** -key $Key

Пример – за CredentialObject:

$User = "MyUserName"

$PasswordFile = "\\Machine1\SharedPath\Password.txt"

$KeyFile = "\\Machine1\SharedPath\AES.key"

$key = **Get-Content** $KeyFile

$MyCredential = **New-Object** -TypeName System.Management.Automation.PSCredential `

 -ArgumentList $User, (**Get-Content** $PasswordFile | **ConvertTo-SecureString** -Key $key)

# Използвайки Self-signed certificate

Как да създадем Credential обект?

$svcAccountName = **Read-Host** "Write what will be the service account using this certificate"

$sharedPath = **Read-Host** "Write where the password and username files are stored. Do it in the format C:\temp\ (not C:\temp ) or \\server-01\shared\ (not \\server-01\shared )”

$upnFilePath = $sharedPath+$svcAccountName+’-upn.txt’

$pwFilePath = $sharedPath+$svcAccountName+'-pw.txt'

function **Get-SvcAccountCredential**

{

    [**OutputType**([string])]

    [**CmdletBinding**()]

*# Retrieve the installed certificate*

    $ImportedCert = **Get-ChildItem** cert:\currentuser\my | **Where-Object** {$\_.Subject -match "$($svcAccountName+’-PSScriptCipherCert’)"}

*# Convert certificate subject to cn= format.*

$certCn =  "cn=$($svcAccountName+’-PSScriptCipherCert’)"

    $EncryptedPwd = **Get-Content** -Path $pwFilePath

*# Decrypt password*

    $DecryptedPwd = $EncryptedPwd | **Unprotect-CmsMessage** -To $certCn

    return $DecryptedPwd

} *# end function*

$svcAccountUpn = **Get-Content** -Path $upnFilePath

$svcAccountPassword = **Get-SvcAccountCredential**

**Write-output** "Constructing credential set to use in PowerShell commands and scripts"  -Verbose

$SecurePwd = $svcAccountPassword | **ConvertTo-SecureString** -AsPlainText -Force

$Cred = **New-Object** System.Management.Automation.PSCredential ($svcAccountUpn, $SecurePwd)

# Използвайки Windows Data Protection API

Пример – за SecureString:

$pass = **Get-Content** "C:\Temp\Password.txt" | **ConvertTo-SecureString**

Пример – за CredentialObject:

$User = "MyUserName"

$File = "C:\Temp\Password.txt"

$MyCredential = **New-Object** -TypeName System.Management.Automation.PSCredential `

    -ArgumentList $User, (**Get-Content** $File | **ConvertTo-SecureString**)

# Използвайки certificate issued by CA

Част 3 – Как да разчетем SecureString

Когато имаме SecureString и искаме да видим в Clear Text какво е съдържанието му:

1 начин:

$SecurePassword = **ConvertTo-SecureString** $PlainPassword

$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($SecurePassword)

$UnsecurePassword = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)

Или

1 начин - 2ри вариант:

$SecurePassword = **ConvertTo-SecureString** $PlainPassword

[Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::SecureStringToBSTR($secureString))

Или

2 начин:

$cred = **Get-Credential**

$svcAccountPassword = $cred.GetNetworkCredential().Password