CryptSharp Namespace

CryptSharp provides a number of password crypt algorithms - BCrypt, LDAP, MD5 (and Apache's hpasswd variant), PHPass (WordPress, phpBB, Drupal), SHA256, SHA512, and Traditional and Extended DES. Additionally it includes Blowfish, SCrypt, and PBKDF2 for any HMAC (.NET's built-in PBKDF2 implementation supports only SHA-1).

If you are looking to store passwords, odds are, CryptSharp has the algorithms you need.

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlowfishCrypter</td>
<td>Blowfish crypt, sometimes called BCrypt. A very good choice.</td>
</tr>
<tr>
<td>Crypter</td>
<td>Crypts and verifies passwords. The main class for most uses of this library.</td>
</tr>
<tr>
<td>CrypterEnvironment</td>
<td>Lets you customize the list of crypt algorithms your program will accept.</td>
</tr>
<tr>
<td>CrypterOption</td>
<td>Options that modify the crypt operation.</td>
</tr>
<tr>
<td>CrypterOptionKey</td>
<td>The key type for options.</td>
</tr>
<tr>
<td>CrypterOptions</td>
<td>Stores options for the crypt operation.</td>
</tr>
<tr>
<td>CrypterProperty</td>
<td>Properties inherent to particular crypt algorithms.</td>
</tr>
<tr>
<td>ExtendedDesCrypter</td>
<td>Extended DES crypt.</td>
</tr>
<tr>
<td>LdapCrypter</td>
<td>LDAP schemes such as {SHA}.</td>
</tr>
<tr>
<td>LdapCrypterOption</td>
<td>Options that modify the LDAP crypt operation.</td>
</tr>
<tr>
<td>MD5Crypter</td>
<td>MD5 crypt, supported by nearly all systems. A variant supports Apache hpasswd files.</td>
</tr>
<tr>
<td>Sha256Crypter</td>
<td>SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.</td>
</tr>
<tr>
<td>Sha512Crypter</td>
<td>SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.</td>
</tr>
<tr>
<td>ShaCrypter</td>
<td>Base class for Sha256Crypter and Sha512Crypter.</td>
</tr>
<tr>
<td>Enumeration</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BlowfishCrypterVariant</td>
<td>Variations of the Blowfish crypt algorithm. You only need concern yourself with Blowfish crypt variations if you have passwords generated pre-2011 using the C-language crypt_blowfish library or a port thereof. CryptSharp was implemented from specification and is not a port, and therefore never had the bug these variants pertain to.</td>
</tr>
<tr>
<td>LdapCrypterVariant</td>
<td>LDAP password schemes.</td>
</tr>
<tr>
<td>MD5CrypterVariant</td>
<td>Modified versions of the MD5 crypt algorithm.</td>
</tr>
<tr>
<td>PhpassCrypterVariant</td>
<td>Modified versions of the PHPass crypt algorithm.</td>
</tr>
</tbody>
</table>
BlowfishCrypter Class

Blowfish crypt, sometimes called BCrypt. A very good choice.

Inheritance Hierarchy

System
  Object
  CryptSharp
    CryptSharpCrypter
  CryptSharp
  BlowfishCrypter

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>public class BlowfishCrypter : Crypter</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The BlowfishCrypter type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlowfishCrypter</td>
<td>Initializes a new instance of the BlowfishCrypter class</td>
</tr>
</tbody>
</table>

Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted</td>
</tr>
</tbody>
</table>
Passwords take the form `algorithm+salt+hash`, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrrides `Crypter.Crypt(Byte, String)`.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Crypt(Byte, CrypterOptions)</code></td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><code>Crypt(String, CrypterOptions)</code></td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from <code>Crypter</code>.)</td>
</tr>
</tbody>
</table>
| `Crypt(String, String)` | Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using `GenerateSalt(CrypterOptions)`.
Because crypted passwords take the form `algorithm+salt+hash`, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from `Crypter`.) |
| `GenerateSalt` | Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from `Crypter`.) |
| `GenerateSalt(Int32)` | Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from `Crypter`.) |
| `GenerateSalt(CrypterOptions)` | Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form `algorithm+salt+hash`, so the salt is always carried along with the crypted password). |
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**BlowfishCrypter Constructor**  
CryptSharp

Initializes a new instance of the **BlowfishCrypter** class

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
C# | VB | C++ | F# | Copy

public BlowfishCrypter()
```

### See Also

Reference

- BlowfishCrypter Class
- CryptSharp Namespace

_Copyright 2010-2014 James F. Bellinger_  
_Send comments on this topic to jfb@zer7.com_
# BlowfishCrypter Methods

The **BlowfishCrypter** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GenerateSalt</strong></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(Int32)</strong></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(CrypterOptions)</strong></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
BlowfishCrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**BlowfishCrypterCanCrypt Method**

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public override bool CanCrypt(
    string salt
)
```

### Parameters

**salt**
Type: `System.String`  
The salt string or crypted password.

### Return Value

Type: `Boolean`  
`true` if the algorithm is compatible.

## See Also

**Reference**

BlowfishCrypter Class  
CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
# BlowfishCrypterCrypt Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form \texttt{algorithm+salt+hash}, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form \texttt{algorithm+salt+hash}, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- BlowfishCrypter Class
- CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**
Send comments on this topic to jfb@zer7.com
BlowfishCrypterCrypt Method (Byte, String)  

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using `GenerateSalt(CrypterOptions)`. Because crypted passwords take the form `algorithm+salt+hash`, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public override string Crypt(  
    byte[] password,  
    string salt  
)
```

**Parameters**

- `password`  
  Type: `SystemByte`  
  The bytes of the password.

- `salt`  
  Type: `SystemString`  
  The salt string or crypted password containing a salt string.

**Return Value**

- Type: `String`  
  The crypted password.

### See Also

- **Reference**
  - BlowfishCrypter Class
  - Crypt Overload
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- BlowfishCrypter Class
- CryptSharp Namespace

---

*Copyright 2010-2014 James F. Bellinger*

Send comments on this topic to jfb@zer7.com
BlowfishCrypterGenerateSalt Method (CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form `algorithm+salt+hash`, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override string GenerateSalt(
    CrypterOptions options
)
```

**Parameters**

- `options`
  - Type: CryptSharpCrypterOptions
  - Options modifying the salt generation.

**Return Value**

- Type: String
  - The salt string.

### See Also

**Reference**

- BlowfishCrypter Class
- GenerateSalt Overload
- CryptSharp Namespace
The **BlowfishCrypter** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

## See Also

Reference

- **BlowfishCrypter Class**
- **CryptSharp Namespace**

**Copyright 2010-2014 James F. Bellinger**
Send comments on this topic to jfb@zer7.com
BlowfishCrypterProperties Property

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public override CrypterOptions Properties { get; }
```

Property Value
Type: CrypterOptions

See Also
Reference
BlowfishCrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**BlowfishCrypterVariant Enumeration**  

Variations of the Blowfish crypt algorithm. You only need concern yourself with Blowfish crypt variations if you have passwords generated pre-2011 using the C-language crypt_blowfish library or a port thereof. CryptSharp was implemented from specification and is not a port, and therefore never had the bug these variants pertain to.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
c# | VB | C++ | F#    
---|----|-----|-------
public enum BlowfishCrypterVariant
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unspecified</td>
<td>0</td>
<td>The $2a$ prefix indicates nothing about whether or not the crypted password was created with a pre-2011 version of the C-language crypt_blowfish library. Pre-2011, that library had a sign extension bug affecting non-ASCII passwords. See EmulateCryptBlowfishSignExtensionBug for a more detailed explanation of the bug in question.</td>
</tr>
<tr>
<td>Compatible</td>
<td>1</td>
<td>The $2x$ prefix indicates that these passwords were generated with pre-2011 crypt_blowfish or a port originating from it. If you have old crypted non-ASCII passwords you can't re-derive, and still want to verify them with CryptSharp, ensure that they have the $2x$ prefix instead of the $2a$ prefix. This will indicate to CryptSharp that it should emulate the bug when verifying the password.</td>
</tr>
<tr>
<td>Corrected</td>
<td>2</td>
<td>The $2y$ prefix indicates that pre-2011 crypt_blowfish's sign extension bug does not affect these crypted passwords. For passwords crypted with CryptSharp, this has always been true and as such selecting this variant changes the prefix but otherwise does not affect the output.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**  
CryptSharp Namespace
Crypter Class

Crupts and verifies passwords. The main class for most uses of this library.

Inheritance Hierarchy

SystemObject  CryptSharpCrypter

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

C# | VB | C++ | F# | Copy

```csharp
public abstract class Crypter
```

The Crypter type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password.</td>
</tr>
<tr>
<td>CheckPassword(Byte, String)</td>
<td>Checks if the crypted password matches the given password bytes.</td>
</tr>
<tr>
<td>CheckPassword(String, String)</td>
<td>Checks if the crypted password matches the given password string.</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes.</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string.</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password.</td>
</tr>
</tbody>
</table>
Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

Crypt(String, CrypterOptions)

Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.

Crypt(String, String)

Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

GenerateSalt

Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.

GenerateSalt(Int32)

Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.

GenerateSalt(CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

GetCrypter

Searches for a crypt algorithm compatible with the specified crypted password or prefix.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blowfish</td>
<td>Blowfish crypt, sometimes called BCrypt. A very good choice.</td>
</tr>
</tbody>
</table>
### Examples

#### Crypting a Password

```csharp
using CryptSharp;

// Crypt using the Blowfish crypt ("BCrypt") algorithm.
string cryptedPassword = Crypter.Blowfish.Crypt(password);
```

#### Checking a Password

```csharp
using CryptSharp;

// Do the passwords match?
// You can also check a password using the Crypt method.
bool matches = Crypter.CheckPassword(testPassword, cryptedPassword);
```

#### Specifying Options

```csharp
using CryptSharp;

// Specify the $apr1$ Apache htpasswd variant of the MD5 crypt algorithm.
string cryptedPassword = Crypter.MD5.Crypt(password, Crypter.MD5.Apr1);
```
{ CrypterOption.Variant, MD5CrypterVariant.A };

See Also

Reference
CryptSharp Namespace

Inheritance Hierarchy

SystemObject
CryptSharpCrypter
    CryptSharpBlowfishCrypter
    CryptSharpExtendedDesCrypter
    CryptSharpLdapCrypter
    CryptSharpMD5Crypter
    CryptSharpPhpassCrypter
    CryptSharpShaCrypter
    CryptSharpTraditionalDesCrypter

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Crypter Methods

The Crypter type exposes the following members.

Method

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password.</td>
</tr>
<tr>
<td>CheckPassword(Byte, String)</td>
<td>Checks if the crypted password matches the given password bytes.</td>
</tr>
<tr>
<td>CheckPassword(String, String)</td>
<td>Checks if the crypted password matches the given password string.</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes.</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string.</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords.</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords.</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of</td>
</tr>
</tbody>
</table>
The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GenerateSalt(Int32)</strong></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password.</td>
</tr>
<tr>
<td><strong>GenerateSalt(CrypterOptions)</strong></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password).</td>
</tr>
<tr>
<td><strong>GetCrypter</strong></td>
<td>Searches for a crypt algorithm compatible with the specified crypted password or prefix.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**
- **Crypter Class**
- **CryptSharp Namespace**

**Copyright 2010-2014 James F. Bellinger**
Send comments on this topic to JFB@zer7.com
**CrypterCanCrypt Method**

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public abstract bool CanCrypt(string salt)
```

#### Parameters

- **salt**  
  Type: `System.String`  
  The salt string or crypted password.

#### Return Value

Type: `Boolean`  
true if the algorithm is compatible.

### See Also

**Reference**  
Crypter Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
### Crypter CheckPassword Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Crypter icon] CheckPassword(Byte, String)</td>
<td>Checks if the crypted password matches the given password bytes.</td>
</tr>
<tr>
<td>![Crypter icon] CheckPassword(String, String)</td>
<td>Checks if the crypted password matches the given password string.</td>
</tr>
</tbody>
</table>

#### See Also

**Reference**

- Crypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
Crypter

CheckPassword Method (Byte, String)

Checks if the crypted password matches the given password bytes.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static bool CheckPassword(
    byte[] password,
    string cryptedPassword
)
```

**Parameters**

- `password`
  - Type: `System.Byte`
  - The password bytes to test.

- `cryptedPassword`
  - Type: `System.String`
  - The crypted password.

**Return Value**

- Type: `Boolean`
  - `true` if the passwords match.

**See Also**

- Crypter Class
- CheckPassword Overload
- CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
Crypter

**CheckPassword Method (String, String)**

Checks if the crypted password matches the given password string.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static bool CheckPassword(
    string password,
    string cryptedPassword
)
```

**Parameters**

`password`  
- Type: `System.String`  
  - The password string to test. Characters are UTF-8 encoded.

`cryptedPassword`  
- Type: `System.String`  
  - The crypted password.

**Return Value**

Type: `Boolean`  
- `true` if the passwords match.

### See Also

**Reference**

Crypter Class  
CheckPassword Overload  
CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
# CrypterCrypt Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes.</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string.</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords.</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- Crypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
CrypterCrypt Method (Byte)  

Creates a one-way password hash (crypted password) from password bytes.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public string Crypt(byte[] password)
```

### Parameters

**password**

Type: System.Byte  
The bytes of the password.

### Return Value

Type: String  
The crypted password.

### See Also

Reference

Crypter Class  
Crypt Overload  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterCrypt Method (String)

Creates a one-way password hash (crypted password) from a password string.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public string Crypt(   string password   )
```

**Parameters**

`password`

Type: `System.String`  
The password string. Characters are UTF-8 encoded.

**Return Value**

Type: `String`  
The crypted password.

### See Also

**Reference**

Crypter Class  
Crypt Overload  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterCrypt Method (Byte, CrypterOptions)**

Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public string Crypt(
    byte[] password,
    CrypterOptions options
)
```

**Parameters**

- **password**
  - Type: System.Byte
  - The bytes of the password.

- **options**
  - Type: CryptSharpCrypterOptions
  - Options modifying the crypt operation.

**Return Value**

- Type: String
  - The crypted password.

**See Also**

**Reference**

- Crypter Class
- Crypt Overload
- CryptSharp Namespace

*Copyright 2010-2014 James F. Bellinger*  
Send comments on this topic to jfb@zer7.com
CrypterCrypt Method (Byte, String)

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public abstract string Crypt(  
    byte[] password,  
    string salt  
)
```

**Parameters**

- **password**
  - Type: SystemByte
  - The bytes of the password.

- **salt**
  - Type: SystemString
  - The salt string or encrypted password containing a salt string.

**Return Value**

- Type: String
  - The encrypted password.

### See Also

- **Reference**
  - Crypter Class
  - Crypt Overload
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Crypter

Crypt Method (String, CrypterOptions)

Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public string Crypt(
    string password,
    CrypterOptions options
)
```

Parameters

password
Type: System.String
The password string. Characters are UTF-8 encoded.

options
Type: CryptSharpCrypterOptions
Options modifying the crypt operation.

Return Value
Type: String
The crypted password.

See Also

Reference
Crypter Class
Crypt Overload
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterCrypt Method (String, String)

Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using
GenerateSalt(CrypterOptions). Because crypted passwords take the form
algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

**Namespace:** CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public string Crypt(
    string password,
    string salt
)
```

**Parameters**

- **password**
  
  Type: SystemString
  
  The password string. Characters are UTF-8 encoded.

- **salt**
  
  Type: SystemString
  
  The salt string or crypted password containing a salt string.

**Return Value**

Type: String

The crypted password.

### See Also

**Reference**

- Crypter Class
- Crypt Overload
# CrypterGenerateSalt Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password).</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- Crypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
**Crypter**

**GenerateSalt Method**

Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public string GenerateSalt()
```

### Return Value

Type: **String**  
The salt string.

### See Also

**Reference**  
Crypter Class  
GenerateSalt Overload  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterGenerateSalt Method (Int32)

Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.

**Namespace:** CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public string GenerateSalt(
    int rounds
)
```

**Parameters**

- **rounds**
  - Type: System.Int32

**Return Value**

- Type: String
  - The salt string.

### See Also

**Reference**

- Crypter Class
- GenerateSalt Overload
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
CrypterGenerateSalt Method
(CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form `algorithm+salt+hash`, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```
public abstract string GenerateSalt(
    CrypterOptions options
)
```

**Parameters**

- `options` Type: CryptSharpCrypterOptions
  Options modifying the salt generation.

**Return Value**

- Type: String
  The salt string.

**See Also**

**Reference**

- Crypter Class
- GenerateSalt Overload
- CryptSharp Namespace
**CrypterGetCrypter Method**

Searches for a crypt algorithm compatible with the specified crypted password or prefix.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static Crypter GetCrypter(
    string cryptedPassword
)
```

**Parameters**

*cryptedPassword*

Type: `System.String`  
The crypted password or prefix.

**Return Value**

Type: `Crypter`  
A compatible crypt algorithm.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>No compatible crypt algorithm was found.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

Crypter Class  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Crypter Properties

The Crypter type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blowfish</td>
<td>Blowfish crypt, sometimes called BCrypt. A very good choice.</td>
</tr>
<tr>
<td>ExtendedDes</td>
<td>Extended DES crypt.</td>
</tr>
<tr>
<td>Ldap</td>
<td>LDAP schemes such as {SHA}.</td>
</tr>
<tr>
<td>MD5</td>
<td>MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.</td>
</tr>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.</td>
</tr>
<tr>
<td>Sha256</td>
<td>SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.</td>
</tr>
<tr>
<td>Sha512</td>
<td>SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.</td>
</tr>
<tr>
<td>TraditionalDes</td>
<td>Traditional DES crypt.</td>
</tr>
</tbody>
</table>

See Also

Reference

Crypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterBlowfish Property**

Blowfish crypt, sometimes called BCrypt. A very good choice.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static BlowfishCrypter Blowfish { get; }
```

- **Property Value**
  - Type: BlowfishCrypter

### See Also

**Reference**

- Crypter Class
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
CrypterExtendedDes Property

Extended DES crypt.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static ExtendedDesCrypter ExtendedDes { get; }
```

Property Value
Type: ExtendedDesCrypter

See Also

Reference
Crypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterLdap Property

LDAP schemes such as {SHA}.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```
public static LdapCrypter Ldap { get; }
```

Property Value
Type: LdapCrypter

See Also

Reference
Crypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Syntax

public static MD5Crypter MD5 { get; }

See Also

Reference

Crypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Crypter::Phpass Property

PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static PhpassCrypter Phpass { get; }
```

### Property Value

**Type:** PhpassCrypter

### See Also

Reference

- Crypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterProperties Property**

Properties inherent to the particular crypt algorithm. These cannot be modified. See *CrypterProperty* for possible keys.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public virtual CrypterOptions Properties { get; }
```

**Property Value**

**Type:** CrypterOptions

### See Also

**Reference**

- Crypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterSha256 Property**

SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static Sha256Crypter Sha256 { get; }
```

**Property Value**

Type: Sha256Crypter

**See Also**

**Reference**

Crypter Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterSha512 Property**

SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static Sha512Crypter Sha512 { get; }
```

**Property Value**

**Type:** Sha512Crypter

### See Also

**Reference**

- Crypter Class  
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
CrypterTraditionalDes Property

Traditional DES crypt.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static TraditionalDesCrypter TraditionalDes {

```

**Property Value**

**Type:** TraditionalDesCrypter

### See Also

**Reference**

Crypter Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterEnvironment Class**  

CryptSharp

Let's you customize the list of crypt algorithms your program will accept.

**Inheritance Hierarchy**

*SystemObject*  
**CryptSharp**  
**CrypterEnvironment**

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

**Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public class CrypterEnvironment
```

The **CrypterEnvironment** type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrypterEnvironment</td>
<td>Initializes a new instance of the <strong>CrypterEnvironment</strong> class</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckPassword(Byte, String)</td>
<td>Checks if the crypted password matches the given password bytes.</td>
</tr>
<tr>
<td>CheckPassword(String, String)</td>
<td>Checks if the crypted password matches the given password string.</td>
</tr>
<tr>
<td>GetCrypter</td>
<td>Searches for a crypt algorithm compatible with the specified crypted password or prefix.</td>
</tr>
<tr>
<td>MakeReadOnly</td>
<td>Prevents future changes to the environment.</td>
</tr>
<tr>
<td>TryGetCrypter</td>
<td>Searches for a crypt algorithm compatible with the specified crypted password or prefix,</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypters</td>
<td>The collection of crypters in this environment.</td>
</tr>
<tr>
<td>Default</td>
<td>The default environment.</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>true if the environment cannot be changed.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
CrypterEnvironment Constructor

Initializes a new instance of the CrypterEnvironment class.

**Namespace:** CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

**C#**

```csharp
public CrypterEnvironment()
```

### See Also

**Reference**

CrypterEnvironment Class
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The CrypterEnvironment type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckPassword(Byte, String)</td>
<td>Checks if the crypted password matches the given password bytes.</td>
</tr>
<tr>
<td>CheckPassword(String, String)</td>
<td>Checks if the crypted password matches the given password string.</td>
</tr>
<tr>
<td>GetCrypter</td>
<td>Searches for a crypt algorithm compatible with the specified crypted password or prefix.</td>
</tr>
<tr>
<td>MakeReadOnly</td>
<td>Prevents future changes to the environment.</td>
</tr>
<tr>
<td>TryGetCrypter</td>
<td>Searches for a crypt algorithm compatible with the specified crypted password or prefix.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**
- CrypterEnvironment Class
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterEnvironmentCheckPassword Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckPassword(Byte, String)</td>
<td>Checks if the crypted password matches the given password bytes.</td>
</tr>
<tr>
<td>CheckPassword(String, String)</td>
<td>Checks if the crypted password matches the given password string.</td>
</tr>
</tbody>
</table>

See Also

Reference

CrypterEnvironment Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterEnvironment**

**CheckPassword Method (Byte, String)**

Checks if the crypted password matches the given password bytes.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

---

### Syntax

C#

```csharp
public bool CheckPassword(
    byte[] password,
    string cryptedPassword
)
```

---

#### Parameters

- **password**  
  Type: SystemByte  
  The password bytes to test.

- **cryptedPassword**  
  Type: SystemString  
  The crypted password.

#### Return Value

Type: Boolean  
true if the passwords match.

---

### See Also

- CrypterEnvironment Class  
- CheckPassword Overload  
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**CrypterEnvironment**

**CheckPassword Method (String, String)**

Checks if the crypted password matches the given password string.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public bool CheckPassword(
    string password,
    string cryptedPassword
)
```

**Parameters**

- **password**  
  Type: System.String  
  The password string to test. Characters are UTF-8 encoded.

- **cryptedPassword**  
  Type: System.String  
  The crypted password.

**Return Value**

Type: Boolean  
true if the passwords match.

### See Also

- CrypterEnvironment Class  
- CheckPassword Overload  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterEnvironment.GetCrypter Method**  
CryptSharp

Searches for a crypt algorithm compatible with the specified crypted password or prefix.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public Crypter GetCrypter(
    string cryptedPassword
)
```

**Parameters**

`cryptedPassword`
- Type: `System.String`
- The crypted password or prefix.

**Return Value**

Type: Crypter  
A compatible crypt algorithm.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>No compatible crypt algorithm was found.</td>
</tr>
</tbody>
</table>

**See Also**

- **Reference**
  - CrypterEnvironment Class  
  - CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
CrypterEnvironmentMakeReadOnly Method

Prevents future changes to the environment.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public CrypterEnvironment MakeReadOnly()
```

Return Value
Type: CrypterEnvironment
The same CrypterEnvironment.

See Also

Reference
- CrypterEnvironment Class
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterEnvironment.TryGetCrypter Method

Searches for a crypt algorithm compatible with the specified crypted password or prefix.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public bool TryGetCrypter(
    string cryptedPassword,
    out Crypter crypter
)
```

### Parameters

- **cryptedPassword**
  - Type: System.String
  - The crypted password or prefix.

- **crypter**
  - Type: CryptSharp.Crypter
  - A compatible crypt algorithm.

### Return Value

- **Type:** Boolean  
  - `true` if a compatible crypt algorithm was found.

### See Also

- CrypterEnvironment Class
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterEnvironment Properties

The CrypterEnvironment type exposes the following members.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypters</td>
<td>The collection of crypters in this environment.</td>
</tr>
<tr>
<td>Default</td>
<td>The default environment.</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>true if the environment cannot be changed.</td>
</tr>
</tbody>
</table>

See Also

Reference

CrypterEnvironment Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterEnvironmentCrypters Property

The collection of crypters in this environment.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public IList<Crypter> Crypters { get; }
```

**Property Value**

Type: `IList<Crypter>`

**See Also**

**Reference**

- CrypterEnvironment Class
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterEnvironmentDefault Property

The default environment.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static CrypterEnvironment Default { get; }
```

Property Value

Type: CrypterEnvironment

See Also

Reference

CrypterEnvironment Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterEnvironment.IsReadOnly Property**  

*true* if the environment cannot be changed.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```
public bool IsReadOnly { get; }
```

**Property Value**  
**Type:** Boolean

### See Also

**Reference**  
CrypterEnvironment Class  
CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**CrypterOption Class**

Options that modify the crypt operation.

**Inheritance Hierarchy**

```
SystemObject  CryptSharpCrypterOption
              CryptSharpLdapCrypterOption
```

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

**Syntax**

```c#
public class CrypterOption
```

The `CrypterOption` type exposes the following members.

**Fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounds</td>
<td>The number of rounds to iterate.</td>
</tr>
<tr>
<td>Variant</td>
<td>The variant of the crypt algorithm to use.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
**CrypterOption Fields**

The **CrypterOption** type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounds</td>
<td>The number of rounds to iterate.</td>
</tr>
<tr>
<td>Variant</td>
<td>The variant of the crypt algorithm to use.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- CrypterOption Class
- CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
**CrypterOptionRounds Field**

The number of rounds to iterate.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public static readonly CrypterOptionKey Rounds
```

**Field Value**

**Type:** CrypterOptionKey

## See Also

**Reference**

CrypterOption Class  
CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**CrypterOptionVariant Field**

The variant of the crypt algorithm to use.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public static readonly CrypterOptionKey Variant
```

### Field Value

**Type:** CrypterOptionKey

### See Also

**Reference**

- CrypterOption Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterOptionKey Class

The key type for options.

Inheritance Hierarchy

System\Object  CryptSharp CrypterOptionKey

Namespace: CryptSharp  
Assembly: CryptSharp (in CryptSharp.dll)  Version: 2.1.0.0

Syntax

C#  |  VB  |  C++  |  F#

```csharp
public class CrypterOptionKey
```

The CrypterOptionKey type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrypterOptionKey</td>
<td>Creates a new option key.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckValue</td>
<td>Throws an exception if the value is incompatible with this option.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object. (Overrides ObjectToString.)</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A description of the option.</td>
</tr>
<tr>
<td>ValueType</td>
<td>The type of the option's value.</td>
</tr>
</tbody>
</table>
CrypterOptionKey Constructor

Creates a new option key.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public CrypterOptionKey(  
    string description,  
    Type valueType  
)
```

### Parameters

**description**
- Type: `System.String`
  - A description of the option.

**valueType**
- Type: `System.Type`
  - The type of the option's value.

## See Also

**Reference**
- CrypterOptionKey Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
The `CrypterOptionKey` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckValue</td>
<td>Throws an exception if the value is incompatible with this option.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>. (Overrides <code>Object.ToString</code>.)</td>
</tr>
</tbody>
</table>

## See Also

- `CrypterOptionKey Class`
- `CryptSharp Namespace`

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterOptionKey.CheckValue Method**

Throws an exception if the value is incompatible with this option.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public void CheckValue(
    Object value
)
```

**Parameters**

- **value**  
  Type: System.Object  
  The value to check.

**See Also**

**Reference**

- CrypterOptionKey Class  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterOptionKeyToString Method

Returns a String that represents the current Object.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public override string ToString()
```

Return Value

Type: String
A String that represents the current Object.

See Also

Reference

CrypterOptionKey Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterOptionKey Properties

The CrypterOptionKey type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A description of the option.</td>
</tr>
<tr>
<td>ValueType</td>
<td>The type of the option's value.</td>
</tr>
</tbody>
</table>

See Also

Reference

CrypterOptionKey Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
CrypterOptionKeyDescription Property

A description of the option.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public string Description { get; }
```

Property Value

Type: String

See Also

Reference

CrypterOptionKey Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterOptionKeyValueType Property**

The type of the option's value.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public Type ValueType { get; }
```

**Property Value**

Type: `Type`

**See Also**

**Reference**

- CrypterOptionKey Class
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterOptions Class**

Stores options for the crypt operation.

**Inheritance Hierarchy**

`System.Object` → `CryptSharp.CrypterOptions`

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```csharp
public class CrypterOptions : IEnumerable<KeyValuePair<CrypterOptionKey, object>>, IEnumerable
```

The `CrypterOptions` type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrypterOptions</td>
<td>Initializes a new instance of the CrypterOptions class</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Sets the value of an option, if the option has not already been set</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears all options.</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Checks if an option is set.</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Returns an enumerator that iterates through all options.</td>
</tr>
<tr>
<td>GetValue&lt;CrypterOptionKey&gt;</td>
<td>Gets the value of an option, if the option is set, or a default value otherwise.</td>
</tr>
<tr>
<td>GetValue&lt;CrypterOptionKey, T&gt;</td>
<td>Gets the value of an option, if the option is set, or a specified default value otherwise.</td>
</tr>
<tr>
<td>MakeReadOnly</td>
<td>Prevents future changes to the options.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>Count</td>
<td>The number of options that have been set.</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td>true if the options cannot be changed.</td>
</tr>
<tr>
<td>Item</td>
<td>Gets or sets an option.</td>
</tr>
<tr>
<td>None</td>
<td>No options.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

**CryptSharp Namespace**

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterOptions Constructor

Initializes a new instance of the CrypterOptions class

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public CrypterOptions()
```

See Also

Reference

CrypterOptions Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterOptions Methods

The CrypterOptions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Sets the value of an option, if the option has not already been set.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears all options.</td>
</tr>
<tr>
<td>ContainsKey</td>
<td>Checks if an option is set.</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Returns an enumerator that iterates through all options.</td>
</tr>
<tr>
<td>GetValue(T(\text{CrypterOptionKey}))</td>
<td>Gets the value of an option, if the option is set, or a default value otherwise.</td>
</tr>
<tr>
<td>GetValue(T(\text{CrypterOptionKey},\ T))</td>
<td>Gets the value of an option, if the option is set, or a specified default value otherwise.</td>
</tr>
<tr>
<td>MakeReadOnly</td>
<td>Prevents future changes to the options.</td>
</tr>
<tr>
<td>Remove</td>
<td>Clears an option.</td>
</tr>
<tr>
<td>TryGetValue</td>
<td>Gets the value of an option, if the option is set.</td>
</tr>
</tbody>
</table>

Top

See Also

Reference

CrypterOptions Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterOptionsAdd Method**

Sets the value of an option, if the option has not already been set.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

**Syntax**

```csharp
public void Add(
    CrypterOptionKey key,
    Object value
)
```

**Parameters**

`key`  
- Type: CryptSharp.CrypterOptionKey  
  - The key of the option.

`value`  
- Type: System.Object  
  - The value of the option.

**See Also**

**Reference**

- CrypterOptions Class  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterOptions.Clear Method

Clears all options.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

[C#]  
```csharp
public void Clear()
```

## See Also

**Reference**  
CrypterOptions Class  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterOptionsContainsKey Method**

Checks if an option is set.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public bool ContainsKey(CrypterOptionKey key)
```

**Parameters**

`key`

Type: CryptSharpCrypterOptionKey  
The key of the option.

**Return Value**

Type: Boolean

`true` if the option is set.

### See Also

**Reference**

CrypterOptions Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterOptionsGetEnumerator Method**

Returns an enumerator that iterates through all options.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public IEnumerator<
    KeyValuePair<CrypterOptionKey, Object>
>
```

**Return Value**  
Type: `IEnumerator<
    KeyValuePair<CrypterOptionKey, Object>
>`. An enumerator for the options.

**Implements**  
`IEnumerable<TGet Enumerator`

## See Also

**Reference**  
- CrypterOptions Class  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterOptionsGetValue Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetValueT(CrypterOptionKey)</td>
<td>Gets the value of an option, if the option is set, or a default value otherwise.</td>
</tr>
<tr>
<td>GetValueT(CrypterOptionKey, T)</td>
<td>Gets the value of an option, if the option is set, or a specified default value otherwise.</td>
</tr>
</tbody>
</table>

See Also

Reference

CrypterOptions Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterOptionsGetValueT Method
(CrypterOptionKey)

Gets the value of an option, if the option is set, or a default value otherwise.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public T GetValue<T>(
    CrypterOptionKey key
)
```

Parameters

- **key**
  - Type: CryptSharpCrypterOptionKey
    - The key of the option.

Type Parameters

- **T**
  - The type of the option's value.

Return Value

- Type: T
  - The option's value.

See Also

- Reference
  - CrypterOptions Class
  - GetValue Overload
  - CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**CrypterOptionsGetValueT Method (CrypterOptionKey, T)**

Gets the value of an option, if the option is set, or a specified default value otherwise.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public T GetValue<T>(
    CrypterOptionKey key,
    T defaultValue
)
```

**Parameters**

- **key**  
  Type: `CryptSharpCrypterOptionKey`  
  The key of the option.

- **defaultValue**  
  Type: `T`  
  The default value if the option is not set.

**Type Parameters**

- **T**  
  The type of the option's value.

**Return Value**

Type: `T`  
The option's value.

**See Also**

- **Reference**  
  - CrypterOptions Class  
  - GetValue Overload  
  - CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
CrypterOptions MakeReadOnly Method

Prevents future changes to the options.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public CrypterOptions MakeReadOnly()
```

**Return Value**
**Type:** CrypterOptions  
The same CrypterOptions.

### See Also

**Reference**

CrypterOptions Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterOptionsRemove Method**

Clears an option.

**Namespace:** CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public bool Remove(
      CrypterOptionKey key
)
```

### Parameters

**key**
- Type: `CryptSharp.CrypterOptionKey`
- The key of the option.

### Return Value

**Type:** Boolean

**true** if the option was found and cleared.

## See Also

**Reference**
- CrypterOptions Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
CrypterOptions.TryGetValu​e Method

Gets the value of an option, if the option is set.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```
public bool TryGetValue(
    CrypterOptionKey key,
    out Object value
)
```

### Parameters

- **key**
  - Type: CrypterOptionKey
  - The key of the option.

- **value**
  - Type: System.Object
  - The value, or null if the option is not set.

### Return Value

- **Type:** Boolean
  - true if the option is set.

## See Also

- CrypterOptions Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
## CrypterOptions Properties

The `CrypterOptions` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>The number of options that have been set.</td>
</tr>
<tr>
<td>IsReadOnly</td>
<td><code>true</code> if the options cannot be changed.</td>
</tr>
<tr>
<td>Item</td>
<td>Gets or sets an option.</td>
</tr>
<tr>
<td>None</td>
<td>No options.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

- CrypterOptions Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
**CrypterOptions.Count Property**

The number of options that have been set.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public int Count { get; }
```

**Property Value**

**Type:** Int32

### See Also

**Reference**

- CrypterOptions Class
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
CrypterOptions.IsReadOnly Property

`true` if the options cannot be changed.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public bool IsReadOnly { get; }
```

**Property Value**
Type: **Boolean**

**See Also**

**Reference**
- CrypterOptions Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterOptionsItem Property**  
CryptSharp

Gets or sets an option.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public Object this[CrypterOptionKey key] { get; set; }
```

**Parameters**

- **key**
  - Type: CryptSharpCrypterOptionKey
  - The key of the option.

**Return Value**

- Type: Object
  - The value of the option.

### See Also

**Reference**

CrypterOptions Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterOptions

No options.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static CrypterOptions None { get; }
```

**Property Value**

Type: CrypterOptions

**See Also**

**Reference**

CrypterOptions Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
CrypterProperty Class

Properties inherent to particular crypt algorithms.

Inheritance Hierarchy

```
System
  Object
  CryptSharp
  CrypterProperty
```

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```
public class CrypterProperty
```

The CrypterProperty type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxPasswordLength</td>
<td>The maximum password length. Bytes beyond this length will have no effect.</td>
</tr>
<tr>
<td>MaxRounds</td>
<td>The maximum number for Rounds.</td>
</tr>
<tr>
<td>MinRounds</td>
<td>The minimum number for Rounds.</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The **CrypterProperty** type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxPasswordLength</td>
<td>The maximum password length. Bytes beyond this length will have no effect.</td>
</tr>
<tr>
<td>MaxRounds</td>
<td>The maximum number for Rounds.</td>
</tr>
<tr>
<td>MinRounds</td>
<td>The minimum number for Rounds.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- **CrypterProperty Class**
- **CryptSharp Namespace**

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
**CrypterPropertyMaxPasswordLength Field** CryptSharp

The maximum password length. Bytes beyond this length will have no effect.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
<th>Copy</th>
</tr>
</thead>
</table>

```csharp
public static readonly CrypterOptionKey MaxPasswordLength
```

**Field Value**  
**Type:** CrypterOptionKey

### See Also

**Reference**  
CrypterProperty Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterPropertyMaxRounds Field**

The maximum number for Rounds.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```csharp
public static readonly CrypterOptionKey MaxRounds
```

**Field Value**  
**Type:** CrypterOptionKey

### See Also

**Reference**  
- CrypterProperty Class  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**CrypterProperty.MinRounds Field**

The minimum number for Rounds.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```
public static readonly CrypterOptionKey MinRounds
```

**Field Value**

Type: CrypterOptionKey

**See Also**

Reference  
CrypterProperty Class  
CryptSharp Namespace

---

*Copyright 2010-2014 James F. Bellinger*  
Send comments on this topic to jfb@zer7.com
**ExtendedDesCrypter Class**

Extended DES crypt.

**Inheritance Hierarchy**

```
System
  SystemObject
    CryptSharpCrypter
      CryptSharpExtendedDesCrypter
```

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

**Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Copy.png" alt="Copy" /></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```csharp
public class ExtendedDesCrypter : Crypter
```

The `ExtendedDesCrypter` type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ExtendedDesCrypter()</code></td>
<td>Initializes a new instance of the <code>ExtendedDesCrypter</code> class</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `CanCrypt()`                | Checks if the particular crypt algorithm is compatible with the salt string or encrypted password.  
                               | (Overides CrypterCanCrypt(String).)                                        |
| `Crypt(Byte)`               | Creates a one-way password hash (encrypted password) from password bytes.    
                               | (Inherited from Crypter.)                                                   |
| `Crypt(String)`             | Creates a one-way password hash (encrypted password) from a password string. |
                               | (Inherited from Crypter.)                                                   |
| `Crypt(Byte, CrypterOptions)`| Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation.  
<pre><code>                           | (Inherited from Crypter.)                                                   |
</code></pre>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. Randomness in a encrypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same encrypted password. If the salt string differs, the same password will generate a different encrypted password (encrypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the encrypted password). (Overrrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**ExtendedDesCrypter Constructor**

Initializes a new instance of the `ExtendedDesCrypter` class

**Namespace**: CryptSharp  
**Assembly**: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public ExtendedDesCrypter()
```

**See Also**

Reference

- ExtendedDesCrypter Class
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
### ExtendedDesCrypter Methods

The `ExtendedDesCrypter` type exposes the following members.

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides <code>CrypterCanCrypt(String)</code>.)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides <code>CrypterCrypt(Byte, String)</code>.)</td>
</tr>
</tbody>
</table>
See Also

GenerateSalt

Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)

GenerateSalt(Int32)

Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)

GenerateSalt(CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

Reference

ExtendedDesCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**ExtendedDesCrypterCanCrypt Method**

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

### Syntax

```csharp
public override bool CanCrypt(
    string salt
)
```

**Parameters**

**salt**  
Type: `System.String`  
The salt string or crypted password.

**Return Value**

Type: `Boolean`  
`true` if the algorithm is compatible.

### See Also

**Reference**

- ExtendedDesCrypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
# ExtendedDesCrypter Encryption Methods

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
</tbody>
</table>

**See Also**

- ExtendedDesCrypter Class
- CryptSharp Namespace

*Copyright 2010-2014 James F. Bellinger*
Send comments on this topic to jfb@zer7.com
ExtendedDesCrypterCrypt Method (Byte, String)

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form $algorithm+salt+hash$, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public override string Crypt(
    byte[] password,
    string salt
)
```

**Parameters**

`password`
- Type: `System.Byte`  
  The bytes of the password.

`salt`
- Type: `System.String`  
  The salt string or encrypted password containing a salt string.

**Return Value**

Type: `String`  
The encrypted password.

**See Also**

Reference
- ExtendedDesCrypter Class
- Crypt Overload
## ExtendedDesCrypter.GenerateSalt Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides Crypter.GenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

- ExtendedDesCrypter Class
- CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
ExtendedDesCrypterGenerateSalt Method (CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form \texttt{algorithm+salt+hash}, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public override string GenerateSalt(CrypterOptions options)
```

**Parameters**

- **options**
  - Type: `CryptSharp.CrypterOptions`
  - Options modifying the salt generation.

**Return Value**

- **Type:** `String`
  - The salt string.

### See Also

- **Reference**
  - ExtendedDesCrypter Class
  - GenerateSalt Overload
  - CryptSharp Namespace
ExtendedDesCrypter Properties

The ExtendedDesCrypter type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

See Also

Reference

ExtendedDesCrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
ExtendedDesCrypterProperties Property

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public override CrypterOptions Properties { get; }
```

**Property Value**

Type: CrypterOptions

**See Also**

Reference

- ExtendedDesCrypter Class  
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**LdapCrypter Class**

LDAP schemes such as {SHA}.

**Inheritance Hierarchy**

```
System
  Object
  CryptSharp
    Crypter
      LdapCrypter
```

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public class LdapCrypter : Crypter
```

The `LdapCrypter` type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LdapCrypter</strong></td>
<td>Initializes a new instance of the <code>LdapCrypter</code> class. The specified environment is searched for the <code>{CRYPT}</code> password scheme.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanCrypt</strong></td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Overrider <code>CrypterCanCrypt(String)</code>.)</td>
</tr>
<tr>
<td><strong>Crypt(Byte)</strong></td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><strong>Crypt(String)</strong></td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><strong>Crypt(Byte, CrypterOptions)</strong></td>
<td>Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. Randomness in a encrypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same encrypted password. If the salt string differs, the same password will generate a different encrypted password (encrypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried</td>
</tr>
</tbody>
</table>
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)</td>
</tr>
</tbody>
</table>

Top

See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**LdapCrypter Constructor**

Initializes a new instance of the `LdapCrypter` class. The specified environment is searched for the `{CRYPT}` password scheme.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public LdapCrypter(
    CrypterEnvironment cryptSchemeEnvironment
)
```

**Parameters**

- `cryptSchemeEnvironment`  
  - Type: `CryptSharpCrypterEnvironment`  
  - The environment for the `{CRYPT}` password scheme.

### See Also

**Reference**

- LdapCrypter Class  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
LdapCrypter Methods

The LdapCrypter type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. Randomness in a encrypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same encrypted password. If the salt string differs, the same password will generate a different encrypted password (encrypted passwords have the form <em>algorithm+salt+hash</em>, so the salt is always carried along with the encrypted password). (Overrides Crypter.GenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

LdapCrypter Class  
CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
LdapCrypterCanCrypt Method

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public override bool CanCrypt(
    string salt
)
```

### Parameters

- **salt**  
  - Type: `System.String`  
  - The salt string or crypted password.

### Return Value

- Type: `Boolean`  
  - `true` if the algorithm is compatible.

## See Also

**Reference**

- LdapCrypter Class  
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
LdapCrypterCrypt Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
</tbody>
</table>

See Also

Reference

LdapCrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
LdapCrypterCrypt Method (Byte, String)  

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form *algorithm+salt+hash*, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0  

**Syntax**

```csharp
public override string Crypt(byte[] password, string salt)
```

**Parameters**

`password`  
Type: SystemByte  
The bytes of the password.

`salt`  
Type: SystemString  
The salt string or crypted password containing a salt string.

**Return Value**

Type: String  
The crypted password.

**See Also**

**Reference**

LdapCrypter Class  
Crypt Overload
## LdapCrypterGenerateSalt Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides LdapCrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

**Top**

### See Also

Reference

LdapCrypter Class

CryptSharp Namespace
LdapCrypterGenerateSalt Method (CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form \texttt{algorithm+salt+hash}, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override string GenerateSalt(CrypterOptions options)
```

**Parameters**

- `options`  
  Type: CryptSharp.CrypterOptions  
  Options modifying the salt generation.

**Return Value**

Type: String  
The salt string.

### See Also

- **Reference**
  - LdapCrypter Class
  - GenerateSalt Overload
  - CryptSharp Namespace
The **LdapCrypter** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See <strong>CrypterProperty</strong> for possible keys. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
</tbody>
</table>

---

**See Also**

Reference

- **LdapCrypter Class**
- **CryptSharp Namespace**

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to [jfb@zer7.com](mailto:jfb@zer7.com)
**LdapCrypterOption Class**

Options that modify the LDAP crypt operation.

**Inheritance Hierarchy**

System\Object  CryptSharpCrypterOption  CryptSharpLdapCrypterOption

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

**Syntax**

```csharp
public class LdapCrypterOption : CrypterOption
```

The **LdapCrypterOption** type exposes the following members.

**Fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypter</td>
<td>The crypter to use with Crypt.</td>
</tr>
<tr>
<td>CrypterOptions</td>
<td>The options to pass to the crypter specified by Crypter.</td>
</tr>
</tbody>
</table>

**See Also**

Reference  
**CryptSharp Namespace**

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
The `LdapCrypterOption` type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Crypter</code></td>
<td>The crypter to use with <code>Crypt</code>.</td>
</tr>
<tr>
<td><code>CrypterOptions</code></td>
<td>The options to pass to the crypter specified by <code>Crypter</code>.</td>
</tr>
</tbody>
</table>

[See Also](#)

[LdapCrypterOption Class](#)

[CryptSharp Namespace](#)

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
LdapCrypterOptionCrypter Field

The crypter to use with Crypt.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public static readonly CrypterOptionKey Crypter
```

**Field Value**  
**Type:** CrypterOptionKey

### See Also

**Reference**  
LdapCrypterOption Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
LdapCrypterOption.CrypterOptions Field

The options to pass to the crypter specified by Crypter.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static readonly CrypterOptionKey CrypterOptions
```

**Field Value**

**Type:** CrypterOptionKey

### See Also

**Reference**

LdapCrypterOption Class  
CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
LdapCrypterVariant Enumeration

LDAP password schemes.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public enum LdapCrypterVariant
```

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSha</td>
<td>0</td>
<td>Salted SHA-1. This is the default.</td>
</tr>
<tr>
<td>Sha</td>
<td>1</td>
<td>Unsalted SHA-1. Used in htpasswd files.</td>
</tr>
<tr>
<td>SSha256</td>
<td>6</td>
<td>Salted SHA-256.</td>
</tr>
<tr>
<td>Sha256</td>
<td>7</td>
<td>Unsalted SHA-256.</td>
</tr>
<tr>
<td>SSha384</td>
<td>8</td>
<td>Salted SHA-384.</td>
</tr>
<tr>
<td>Sha384</td>
<td>9</td>
<td>Unsalted SHA-384.</td>
</tr>
<tr>
<td>SSha512</td>
<td>10</td>
<td>Salted SHA-512.</td>
</tr>
<tr>
<td>Sha512</td>
<td>11</td>
<td>Unsalted SHA-512.</td>
</tr>
<tr>
<td>SMD5</td>
<td>2</td>
<td>Salted MD-5.</td>
</tr>
<tr>
<td>MD5</td>
<td>3</td>
<td>Unsalted MD-5.</td>
</tr>
<tr>
<td>Cleartext</td>
<td>4</td>
<td>No crypt operation is performed. The password can be read easily.</td>
</tr>
<tr>
<td>Crypt</td>
<td>5</td>
<td>Any crypt algorithm. If you specify this for Variant, you must also set Crypter and may optionally set CrypterOptions.</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**MD5Crypter Class**

MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.

**Inheritance Hierarchy**

```
System
  Object
  CryptSharpCrypter
    CryptSharpMD5Crypter
```

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public class MD5Crypter : Crypter
```

The **MD5Crypter** type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5Crypter</td>
<td>Initializes a new instance of the <strong>MD5Crypter</strong> class</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation.</td>
</tr>
</tbody>
</table>

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0
Crypt(String, CrypterOptions)

Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation.
(Inherited from Crypter.)

Crypt(String, String)

Creates a one-way password hash (encrypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form `algorithm+salt+hash`, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.
(Inherited from Crypter.)

Crypt(Byte, String)

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form `algorithm+salt+hash`, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.
(Overrides CrypterCrypt(Byte, String).)

GenerateSalt

Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password.
(Inherited from Crypter.)

GenerateSalt(Int32)

Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password.
(Inherited from Crypter.)

GenerateSalt(CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. Randomness in a encrypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same encrypted password. If the salt string differs, the same password will generate a different encrypted password (encrypted passwords have the form `algorithm+salt+hash`, so the salt is always carried
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**MD5Crypter Constructor**

Initializes a new instance of the **MD5Crypter** class

**Namespace**: CryptSharp

**Assembly**: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```csharp
public MD5Crypter()
```

### See Also

**Reference**

MD5Crypter Class

CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
The **MD5Crypter** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

MD5Crypter Class  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**MD5CrypterCanCrypt Method**

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override bool CanCrypt(string salt)
```

**Parameters**

- **salt**
  - Type: `System.String`
  - The salt string or crypted password.

**Return Value**

- Type: `Boolean`
- `true` if the algorithm is compatible.

### See Also

- **Reference**
  - MD5Crypter Class
  - CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
# MD5CrypterCrypt Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
</tbody>
</table>

## See Also

**MD5Crypter Class**

**CryptSharp Namespace**

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
MD5CrypterCrypt Method (Byte, String)  

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using `GenerateSalt(CrypterOptions)`. Because encrypted passwords take the form `algorithm+salt+hash`, if you pass an encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override string Crypt(
    byte[] password,
    string salt
)
```

**Parameters**

- **password**
  - Type: `System.Byte`
  - The bytes of the password.

- **salt**
  - Type: `System.String`
  - The salt string or encrypted password containing a salt string.

**Return Value**

- **Type:** `String`
- The encrypted password.

### See Also

**Reference**

- MD5Crypter Class
- Crypt Overload
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
# MD5CrypterGenerateSalt Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="GenerateSalt" /></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><img src="image" alt="GenerateSalt(Int32)" /></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><img src="image" alt="GenerateSalt(CrypterOptions)" /></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

## See Also

### Reference

- **MD5Crypter Class**
- **CryptSharp Namespace**

---

*Copyright 2010-2014 James F. Bellinger*

Send comments on this topic to jfb@zer7.com
Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form `algorithm+salt+hash`, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override string GenerateSalt(
    CrypterOptions options
)
```

**Parameters**

- `options`  
  Type: CryptSharp.CrypterOptions  
  Options modifying the salt generation.

**Return Value**

- Type: `String`  
  The salt string.

### See Also

**Reference**

- MD5Crypter Class  
- GenerateSalt Overload  
- CryptSharp Namespace
MD5Crypter Properties

The MD5Crypter type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)</td>
</tr>
</tbody>
</table>

See Also

Reference

MD5Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**MD5CrypterVariant Enumeration**

Modified versions of the MD5 crypt algorithm.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public enum MD5CrypterVariant
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>0</td>
<td>Standard MD5 crypt.</td>
</tr>
<tr>
<td>Apache</td>
<td>1</td>
<td>Apache htpasswd files have a different prefix. Due to the nature of MD5 crypt, this also affects the crypted password.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**  
CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**PhpassCrypter Class**

PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+

**Inheritance Hierarchy**

```
System.Object  CryptSharpCrypter  CryptSharpPhpassCrypter
```

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```csharp
public class PhpassCrypter : Crypter
```

The **PhpassCrypter** type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhpassCrypter</td>
<td>Initializes a new instance of the <strong>PhpassCrypter</strong> class</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Overrices CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Crypt(String, CrypterOptions)</strong></td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>Crypt(String, String)</strong></td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>Crypt(Byte, String)</strong></td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td><strong>GenerateSalt</strong></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(Int32)</strong></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(CrypterOptions)</strong></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). ( Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

### See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
**PhpassCrypter Constructor**

Initializes a new instance of the **PhpassCrypter** class

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public PhpassCrypter()
```

### See Also

**PhpassCrypter Class**  
**CryptSharp Namespace**

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
### PhpassCrypter Methods

The PhpassCrypter type exposes the following members.

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
</tbody>
</table>
Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)

Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form \texttt{algorithm+salt+hash}, so the salt is always carried along with the crypted password). (Overrides Crypter\texttt{GenerateSalt(CrypterOptions)}.)

See Also

Reference

PhpassCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to \texttt{jfb@zer7.com}
**PhpassCrypterCanCrypt Method**  

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override bool CanCrypt(
    string salt
)
```

**Parameters**

- `salt`  
  Type: `System.String`  
  The salt string or crypted password.

**Return Value**

Type: `Boolean`  
`true` if the algorithm is compatible.

### See Also

**Reference**

- PhpassCrypter Class  
- CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overridess CrypterCrypt(Byte, String).)</td>
</tr>
</tbody>
</table>

See Also

Reference

PhpassCrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
PhpassCrypter

Crypt Method (Byte, String)  CryptSharp

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override string Crypt(byte[] password, string salt)
```

### Parameters

**password**  
- Type: SystemByte  
  - The bytes of the password.

**salt**  
- Type: SystemString  
  - The salt string or encrypted password containing a salt string.

### Return Value

**Type:** String  
- The encrypted password.

### See Also

**Reference**  
- **PhpassCrypter Class**  
- **Crypt Overload**
## PhpassCrypterGenerateSalt Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <strong>algorithm+salt+hash</strong>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

- PhpassCrypter Class
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
PhpassCrypterGenerateSalt Method (CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form \texttt{algorithm+salt+hash}, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```
public override string GenerateSalt(
    CrypterOptions options
)
```

**Parameters**

- **options**
  - Type: CryptSharpCrypterOptions
  - Options modifying the salt generation.

**Return Value**

- **Type:** String
  - The salt string.

**See Also**

**Reference**

- PhpassCrypter Class
- GenerateSalt Overload
- CryptSharp Namespace
The **PhpassCrypter** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Properties</strong></td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

---

**See Also**

**Reference**

**PhpassCrypter Class**

**CryptSharp Namespace**

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
**PhpassCrypterProperties Property**

Properties inherent to the particular crypt algorithm. These cannot be modified. See `CrypterProperty` for possible keys.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override CrypterOptions Properties { get; }
```

**Property Value**

**Type:** CrypterOptions

### See Also

**Reference**

- PhpassCrypter Class
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**PhpassCrypterVariant Enumeration**

Modified versions of the PHPass crypt algorithm.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public enum PhpassCrypterVariant
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phpbb</td>
<td>1</td>
<td>phpBB changes the prefix but the algorithm is otherwise identical.</td>
</tr>
<tr>
<td>Drupal</td>
<td>2</td>
<td>Drupal 7+ uses SHA512 instead of MD5.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Sha256Crypter Class

SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

Inheritance Hierarchy

System
  Object
  CryptSharp
    Crypter
      ShaCrypter
        Sha256Crypter

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public class Sha256Crypter : ShaCrypter
```

The **Sha256Crypter** type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sha256Crypter</td>
<td>Initializes a new instance of the <strong>Sha256Crypter</strong> class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Inherited from ShaCrypter.)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from ShaCrypter.)</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different...</td>
</tr>
</tbody>
</table>
crypted password (crypted passwords have the form \texttt{algorithm+salt+hash}, so the salt is always carried along with the crypted password). (Inherited from \texttt{ShaCrypter}.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from \texttt{ShaCrypter}.)</td>
</tr>
</tbody>
</table>

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Sha256Crypter Constructor

Initializes a new instance of the Sha256Crypter class

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public Sha256Crypter()
```

See Also

Reference

Sha256Crypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
## Sha256Crypter Methods

The **Sha256Crypter** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Inherited from <strong>ShaCrypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string and a salt string. The salt can be produced using <strong>GenerateSalt(CrypterOptions)</strong>. Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using <strong>GenerateSalt(CrypterOptions)</strong>. Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Inherited from <strong>ShaCrypter</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Inherited from ShaCrypter.)</td>
</tr>
</tbody>
</table>

See Also

Reference

Sha256Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
Sha256Crypter Properties

The Sha256Crypter type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from ShaCrypter.)</td>
</tr>
</tbody>
</table>

See Also

Sha256Crypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Sha512Crypter Class

SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

Inheritance Hierarchy

System
  -> Object
  -> CryptSharp
  -> ShaCrypter
  -> Sha512Crypter

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```csharp
public class Sha512Crypter : ShaCrypter
```

The Sha512Crypter type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sha512Crypter</td>
<td>Initializes a new instance of the Sha512Crypter class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Inherited from ShaCrypter.)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>Crypt(Byte, CrypterOptions)</code></td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><code>Crypt(String, CrypterOptions)</code></td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><code>Crypt(String, String)</code></td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><code>Crypt(Byte, String)</code></td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from <code>ShaCrypter</code>.)</td>
</tr>
<tr>
<td><code>GenerateSalt</code></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from <code>Crypter</code>.)</td>
</tr>
<tr>
<td><code>GenerateSalt(Int32)</code></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from <code>Crypter</code>.)</td>
</tr>
</tbody>
</table>
| `GenerateSalt(CrypterOptions)` | Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different
crypted password (crypted passwords have the form `algorithm+salt+hash`, so the salt is always carried along with the crypted password). (Inherited from `ShaCrypter`.)

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See <code>CrypterProperty</code> for possible keys. (Inherited from <code>ShaCrypter</code>.)</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

- **CryptSharp Namespace**

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
Sha512Crypter Constructor

Initializes a new instance of the Sha512Crypter class

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public Sha512Crypter()
```

### See Also

**Reference**

- Sha512Crypter Class
- CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
The **Sha512Crypter** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Inherited from <strong>ShaCrypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using <strong>GenerateSalt(CrypterOptions)</strong>. Because crypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from <strong>Crypter</strong>.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <strong>GenerateSalt(CrypterOptions)</strong>. Because crypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from <strong>ShaCrypter</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Inherited from ShaCrypter.)</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

Sha512Crypter Class

CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
Sha512Crypter Properties

The Sha512Crypter type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from ShaCrypter.)</td>
</tr>
</tbody>
</table>

See Also

Reference

Sha512Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**ShaCrypter Class**

Base class for Sha256Crypter and Sha512Crypter.

**Inheritance Hierarchy**

```
System\Object  CryptSharp\Crypter
   CryptSharp\ShaCrypter
   CryptSharp\Sha256Crypter
   CryptSharp\Sha512Crypter
```

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public abstract class ShaCrypter : Crypter
```

The **ShaCrypter** type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or encrypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
</tbody>
</table>
Password string and a salt string. The salt can be produced using `GenerateSalt(CrypterOptions)`. Because crypted passwords take the form `algorithm+salt+hash`, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

(Overriden from `Crypter`.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Overriden From</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Crypt(Byte, String)</strong></td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overriden from <code>Crypter</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>GenerateSalt</strong></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from <code>Crypter</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>GenerateSalt(Int32)</strong></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from <code>Crypter</code>).</td>
<td></td>
</tr>
<tr>
<td><strong>GenerateSalt(CrypterOptions)</strong></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overriden from <code>CrypterGenerateSalt(CrypterOptions)</code>).</td>
<td></td>
</tr>
</tbody>
</table>

**Properties**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CryptProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
ShaCrypter Methods

The ShaCrypter type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate and verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate and verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GenerateSalt</strong></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(Int32)</strong></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(CrypterOptions)</strong></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

ShaCrypter Class

CryptSharp Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
ShaCrypterCanCrypt Method

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public override bool CanCrypt(string salt)
```

**Parameters**

`salt`  
Type: `System.String`  
The salt string or crypted password.

**Return Value**

Type: `Boolean`  
`true` if the algorithm is compatible.

**See Also**

Reference  
**ShaCrypter Class**  
**CryptSharp Namespace**

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
## ShaCrypterCrypt Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form \textit{algorithm+salt+hash}, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form \textit{algorithm+salt+hash}, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
</tbody>
</table>

**Top**

### See Also

**Reference**

- ShaCrypter Class
- CryptSharp Namespace

**Copyright 2010-2014 James F. Bellinger**
Send comments on this topic to jfb@zer7.com
ShaCrypter

**Crypt Method (Byte, String)**

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using `GenerateSalt(CrypterOptions)`. Because encrypted passwords take the form `algorithm+salt+hash`, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public override string Crypt(
    byte[] password,
    string salt
)
```

**Parameters**

- **password**
  - Type: `System.Byte`  
  - The bytes of the password.

- **salt**
  - Type: `System.String`  
  - The salt string or encrypted password containing a salt string.

**Return Value**

- Type: `String`  
  - The encrypted password.

### See Also

**Reference**

- [ShaCrypter Class](#)  
- [Crypt Overload](#)
ShacrypterGenerateSalt Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🍊 GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>🍊 GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>🍊 GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

See Also

Reference

Shacrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
ShaCrypter GenerateSalt Method (CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same unencrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

### Syntax

```csharp
public override string GenerateSalt(CrypterOptions options)
```

### Parameters
- **options**  
  Type: CryptSharpCrypterOptions  
  Options modifying the salt generation.

### Return Value
- **Type:** String  
  The salt string.

### See Also
- **Reference**  
  ShaCrypter Class  
  GenerateSalt Overload  
  CryptSharp Namespace
The `ShaCrypter` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Properties</strong></td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See <code>CrypterProperty</code> for possible keys. (Overrides <code>CrypterProperties</code>.)</td>
</tr>
</tbody>
</table>

See Also

**Reference**

- `ShaCrypter Class`
- `CryptSharp Namespace`
ShaCrypterProperties Property

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public override CrypterOptions Properties { get; }
```

Property Value
Type: CrypterOptions

See Also

Reference
ShaCrypter Class
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
TraditionalDesCrypter Class

Traditional DES crypt.

Inheritance Hierarchy

System\Object  CryptSharpCrypter

CryptSharpTraditionalDesCrypter

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

public class TraditionalDesCrypter : Crypter

The TraditionalDesCrypter type exposes the following members.

 Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TraditionalDesCrypter</td>
<td>Initializes a new instance of the TraditionalDesCrypter class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| CanCrypt              | Checks if the particular crypt algorithm is compatible with the salt string or encrypted password.  
(Inherits CrypterCanCrypt(String).) |
| Crypt(Byte)           | Creates a one-way password hash (encrypted password) from password bytes.  
(Inherited from Crypter.) |
| Crypt(String)         | Creates a one-way password hash (encrypted password) from a password string.  
(Inherited from Crypter.) |
| Crypt(Byte, CrypterOptions) | Creates a one-way password hash (encrypted password) from password bytes. Options modify the crypt operation.  
(Inherited from Crypter.) |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Crypt(String, CrypterOptions)</code></td>
<td>Creates a one-way password hash (encrypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><code>Crypt(String, String)</code></td>
<td>Creates a one-way password hash (encrypted password) from a password string and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><code>Crypt(Byte, String)</code></td>
<td>Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because encrypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate <em>and</em> verify encrypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td><code>GenerateSalt</code></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><code>GenerateSalt(Int32)</code></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><code>GenerateSalt(CrypterOptions)</code></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the encrypted password to be different even if two users have the same unencrypted password. Randomness in a encrypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same encrypted password. If the salt string differs, the same password will generate a different encrypted password (encrypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the encrypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)</td>
</tr>
</tbody>
</table>

## See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
**TraditionalDesCrypter Constructor**  
CryptSharp

Initializes a new instance of the `TraditionalDesCrypter` class

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public TraditionalDesCrypter()
```

### See Also

- **Reference**  
  - `TraditionalDesCrypter Class`  
  - CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
## TraditionalDesCrypter Methods

The **TraditionalDesCrypter** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanCrypt</td>
<td>Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)</td>
</tr>
<tr>
<td>Crypt(Byte)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String)</td>
<td>Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, CrypterOptions)</td>
<td>Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(String, String)</td>
<td>Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>Crypt(Byte, String)</td>
<td>Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form <strong>algorithm+salt+hash</strong>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate <em>and</em> verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)</td>
</tr>
<tr>
<td><strong>GenerateSalt</strong></td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(Int32)</strong></td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td><strong>GenerateSalt(CrypterOptions)</strong></td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

TraditionalDesCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
TraditionalDesCrypterCanCrypt Method

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public override bool CanCrypt(
    string salt
)
```

**Parameters**

`salt`

Type: `System.String`  
The salt string or crypted password.

**Return Value**

Type: `Boolean`  
`true` if the algorithm is compatible.

**See Also**

Reference

TraditionalDesCrypter Class  
CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Name | Description
--- | ---
Crypt(Byte) | Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
Crypt(String) | Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
Crypt(Byte, CrypterOptions) | Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
Crypt(String, CrypterOptions) | Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
Crypt(String, String) | Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form \texttt{algorithm+salt+hash}, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
Crypt(Byte, String) | Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form \texttt{algorithm+salt+hash}, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)
Send comments on this topic to jfb@zer7.com
TraditionalDesCrypterCrypt Method (Byte, String)

Creates a one-way password hash (encrypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because encrypted passwords take the form algorithm+salt+hash, if you pass a encrypted password as the salt parameter, the same algorithm and salt will be used to re-encrypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same encrypted password will result. Therefore, this method can both generate *and* verify encrypted passwords.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public override string Crypt(
    byte[] password,
    string salt
)
```

### Parameters

**password**

- **Type:** SystemByte  
  - The bytes of the password.

**salt**

- **Type:** SystemString  
  - The salt string or encrypted password containing a salt string.

### Return Value

- **Type:** String  
  - The encrypted password.

### See Also

**Reference**

- TraditionalDesCrypter Class  
- Crypt Overload
### TraditionalDesCrypter GenerateSalt Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateSalt</td>
<td>Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(Int32)</td>
<td>Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)</td>
</tr>
<tr>
<td>GenerateSalt(CrypterOptions)</td>
<td>Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form <code>algorithm+salt+hash</code>, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)</td>
</tr>
</tbody>
</table>

#### See Also

**Reference**

- TraditionalDesCrypter Class
- CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form \texttt{algorithm+salt+hash}, so the salt is always carried along with the crypted password).

Namespace: CryptSharp
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public override string GenerateSalt(
    CrypterOptions options
)
```

**Parameters**

- **options**
  - Type: CryptSharpCrypterOptions
  - Options modifying the salt generation.

**Return Value**

- Type: String
  - The salt string.

### See Also

Reference

- TraditionalDesCrypter Class
- GenerateSalt Overload
- CryptSharp Namespace
The `TraditionalDesCrypter` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties inherent to the particular crypt algorithm. These cannot be modified. See <code>CrypterProperty</code> for possible keys. (Overrides <code>CrypterProperties</code>.)</td>
</tr>
</tbody>
</table>

### See Also

- `TraditionalDesCrypter Class`
- `CryptSharp Namespace`

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to [jfb@zer7.com](mailto:jfb@zer7.com)
**TraditionalDesCrypterProperties Property**  
CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

**Namespace:** CryptSharp  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override CrypterOptions Properties { get; }
```

**Property Value**  
Type: CrypterOptions

### See Also

**Reference**  
TraditionalDesCrypter Class  
CryptSharp Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
These classes are not directly related to the crypt functionality, but most were necessary for its implementation, so I see no reason not to make them available. You may find them useful for other purposes.

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base16Encoding</td>
<td>Base-16 binary-to-text encodings.</td>
</tr>
<tr>
<td>Base2Encoding</td>
<td>Base-2 binary-to-text encodings.</td>
</tr>
<tr>
<td>Base32Encoding</td>
<td>Base-32 binary to text encodings. I needed multiple variations of base-64 for the various crypt algorithms, and base-16 (hex) for test vectors, so base-32 is mostly a freebie. It's great for e-mail verifications, product keys - really anywhere you need someone to type in a randomly-generated code.</td>
</tr>
<tr>
<td>Base64Encoding</td>
<td>Base-64 binary-to-text encodings.</td>
</tr>
<tr>
<td>BaseEncoding</td>
<td>Performs generic binary-to-text encoding.</td>
</tr>
<tr>
<td>BlowfishCipher</td>
<td>Performs low-level encryption and decryption using the Blowfish cipher.</td>
</tr>
<tr>
<td>DesCipher</td>
<td>Performs low-level encryption and decryption using the DES cipher.</td>
</tr>
<tr>
<td>Pbkdf2</td>
<td>Implements the PBKDF2 key derivation function.</td>
</tr>
<tr>
<td>Salsa20Core</td>
<td>Implements the Salsa20 hash function.</td>
</tr>
<tr>
<td>SCrypt</td>
<td>Implements the SCrypt key derivation function.</td>
</tr>
<tr>
<td>SecureComparison</td>
<td>Provides comparison methods resistant to timing attack.</td>
</tr>
</tbody>
</table>

### Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseEncodingDecodeFilterCallback</td>
<td>A callback to map arbitrary characters onto the characters that can be decoded.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EksBlowfishKeyExpansionFlags</td>
<td>Modifiers for Expensive Key Schedule (EKS) Blowfish</td>
</tr>
</tbody>
</table>
Base16Encoding Class

Base-16 binary-to-text encodings.

Inheritance Hierarchy

System\Object  CryptSharp.Utility\Base16Encoding

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**public static class** Base16Encoding

The **Base16Encoding** type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Hex" /></td>
<td>Hexadecimal base-16 uses the numbers 0-9 for 0-9, and the letters A-F for 10-15.</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The `Base16Encoding` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❓ Hex</td>
<td>Hexadecimal base-16 uses the numbers 0-9 for 0-9, and the letters A-F for 10-15.</td>
</tr>
</tbody>
</table>

See Also

**Reference**

- `Base16Encoding Class`
- `CryptSharp.Utility Namespace`

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Base16EncodingHex Property  

Hexadecimal base-16 uses the numbers 0-9 for 0-9, and the letters A-F for 10-15.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public static BaseEncoding Hex { get; }
```

### Property Value

**Type:** BaseEncoding

### See Also

**Reference**

- Base16Encoding Class
- CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Base2Encoding Class

Base-2 binary-to-text encodings.

Inheritance Hierarchy

System\Object  CryptSharp.Utility\Base2Encoding

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```
public static class Base2Encoding
```

The Base2Encoding type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>Binary. Useful for debugging.</td>
</tr>
</tbody>
</table>

See Also

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The **Base2Encoding** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin</td>
<td>Binary. Useful for debugging.</td>
</tr>
</tbody>
</table>

### See Also

- **Base2Encoding Class**
- **CryptSharp.Utility Namespace**

---

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Base2EncodingBinary Property

Binary. Useful for debugging.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static BaseEncoding Binary { get; }
```

Property Value
Type: BaseEncoding

See Also

Reference

Base2Encoding Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Inheritance Hierarchy

SystemObject  CryptSharp.Utility/Base32Encoding

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static class Base32Encoding
```

The **Base32Encoding** type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s]</td>
<td>Crockford base-32 is somewhat traditional, but still better than the RFC 4648 standard. It is specified at <a href="http://www.crockford.com/wrmg/base32.html">http://www.crockford.com/wrmg/base32.html</a>.</td>
</tr>
<tr>
<td>![s]</td>
<td>ZBase32  z-base-32 is a lowercase base-32 encoding designed to be easily hand-written and read. It is specified at <a href="http://www.zer7.com/files/oss/cryptsharp/zbase32.txt">http://www.zer7.com/files/oss/cryptsharp/zbase32.txt</a>.</td>
</tr>
</tbody>
</table>

See Also

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The **Base32Encoding** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Crockford" /></td>
<td>Crockford base-32 is somewhat traditional, but still better than the RFC 4648 standard. It is specified at <a href="http://www.crockford.com/wrmg/base32.html">http://www.crockford.com/wrmg/base32.html</a>.</td>
</tr>
<tr>
<td><img src="image" alt="ZBase32" /></td>
<td>z-base-32 is a lowercase base-32 encoding designed to be easily hand-written and read. It is specified at <a href="http://www.zer7.com/files/oss/cryptsharp/zbase32.txt">http://www.zer7.com/files/oss/cryptsharp/zbase32.txt</a>.</td>
</tr>
</tbody>
</table>

### See Also

- **Base32Encoding Class**
- **CryptSharp.Utility Namespace**

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
Base32EncodingCrockford Property

Crockford base-32 is somewhat traditional, but still better than the RFC 4648 standard. It is specified at http://www.crockford.com/wrmg/base32.html.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static BaseEncoding Crockford { get; }
```

Property Value
Type: BaseEncoding

See Also

Reference
Base32Encoding Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Base32EncodingZBase32 Property

z-base-32 is a lowercase base-32 encoding designed to be easily hand-written and read. It is specified at http://www.zer7.com/files/oss/cryptsharp/zbase32.txt.

**Namespace:** CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static BaseEncoding ZBase32 { get; }
```

**Property Value**

Type: BaseEncoding

**See Also**

Reference

Base32Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
# Base64Encoding Class

Base-64 binary-to-text encodings.

## Inheritance Hierarchy

System\Object  CryptSharp.Utility\Base64Encoding

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

## Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static class Base64Encoding</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The **Base64Encoding** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blowfish</td>
<td>Blowfish crypt orders characters differently from standard crypt, and begins encoding from the most-significant bit instead of the least-significant bit.</td>
</tr>
<tr>
<td>UnixCrypt</td>
<td>Traditional DES crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.</td>
</tr>
<tr>
<td>UnixMD5</td>
<td>MD5, SHA256, and SHA512 crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

CryptSharp.Utility Namespace

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
## Properties

The `Base64Encoding` type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blowfish</td>
<td>Blowfish crypt orders characters differently from standard crypt, and begins encoding from the most-significant bit instead of the least-significant bit.</td>
</tr>
<tr>
<td>UnixCrypt</td>
<td>Traditional DES crypt base-64, as seen on Unix <code>/etc/passwd</code>, many websites, database servers, etc.</td>
</tr>
<tr>
<td>UnixMD5</td>
<td>MD5, SHA256, and SHA512 crypt base-64, as seen on Unix <code>/etc/passwd</code>, many websites, database servers, etc.</td>
</tr>
</tbody>
</table>

## See Also

- **Reference**
  - `Base64Encoding Class`
  - `CryptSharp.Utility Namespace`

---

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Blowfish crypt orders characters differently from standard crypt, and begins encoding from the most-significant bit instead of the least-significant bit.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static BaseEncoding Blowfish { get; }
```

**Property Value**  
**Type:** BaseEncoding

### See Also

**Reference**  
- Base64Encoding Class  
- CryptSharp.Utility Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
Base64EncodingUnixCrypt Property

Traditional DES crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static BaseEncoding UnixCrypt { get; }
```

**Property Value**  
Type: **BaseEncoding**

### See Also

**Reference**  
- Base64Encoding Class  
- CryptSharp.Utility Namespace
Base64EncodingUnixMD5 Property

MD5, SHA256, and SHA512 crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.

**Namespace:** CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static BaseEncoding UnixMD5 { get; }
```

**Property Value**

**Type:** BaseEncoding

### See Also

**Reference**

Base64Encoding Class

CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
BaseEncoding Class

Performs generic binary-to-text encoding.

Inheritance Hierarchy

System
  Object
  System.Text.Encoding
  CryptSharp.Utility.BaseEncoding

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public class BaseEncoding : Encoding</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The BaseEncoding type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseEncoding(String, Boolean)</td>
<td>Defines a binary-to-text encoding.</td>
</tr>
<tr>
<td>BaseEncoding(String, Boolean, IDictionary&lt;Char, Int32&gt;, BaseEncodingDecodeFilterCallback)</td>
<td>Defines a binary-to-text encoding. Additional decode characters let you add aliases, and a filter callback can be used to make decoding case-insensitive among other things.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetByteCount</td>
<td>When overridden in a derived class, calculates the number of bytes produced by encoding a set of characters from the specified character array. (Overrides Encoding.GetBytes(Char, Int32, Int32).)</td>
</tr>
<tr>
<td>GetBytes(Char, Int32, Byte, Int32)</td>
<td>When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array. (Overrides Encoding.GetBytes(Char, Int32, Int32, Byte, Int32).)</td>
</tr>
<tr>
<td>GetBytes(Char, Int32, Int32, Byte, Int32)</td>
<td>Converts characters from their text representation to a binary</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitMask</td>
<td>The bit mask for a single character in the current encoding.</td>
</tr>
<tr>
<td>BitsPerCharacter</td>
<td>The number of bits per character in the current encoding.</td>
</tr>
<tr>
<td>MsbComesFirst</td>
<td>true if the encoding begins with the most-significant bit of each byte.</td>
</tr>
<tr>
<td></td>
<td>Otherwise, the encoding begins with the least-significant bit.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
### BaseEncoding Constructor

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseEncoding(String, Boolean)</td>
<td>Defines a binary-to-text encoding.</td>
</tr>
<tr>
<td>BaseEncoding(String, Boolean, IDictionaryChar, Int32, BaseEncodingDecodeFilterCallback)</td>
<td>Defines a binary-to-text encoding. Additional decode characters let you add aliases, and a filter callback can be used to make decoding case-insensitive among other things.</td>
</tr>
</tbody>
</table>

---

### See Also

- BaseEncoding Class
- CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BaseEncoding Constructor (String, Boolean) CryptSharp

Defines a binary-to-text encoding.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public BaseEncoding(
    string characterSet,
    bool msbComesFirst
)
```

**Parameters**

- **characterSet**
  - Type: `System.String`
  - The characters of the encoding.

- **msbComesFirst**
  - Type: `System.Boolean`
  - `true` to begin with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.

### See Also

- **Reference**
  - BaseEncoding Class
  - BaseEncoding Overload
  - CryptSharp.Utility Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
BaseEncoding Constructor (String, Boolean, IDictionaryChar, Int32, BaseEncodingDecodeFilterCallback)

Defines a binary-to-text encoding. Additional decode characters let you add aliases, and a filter callback can be used to make decoding case-insensitive among other things.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public BaseEncoding(
    string characterSet,
    bool msbComesFirst,
    IDictionary<char, int> additionalDecodeCharacters,
    BaseEncodingDecodeFilterCallback decodeFilterCallback
)
```

Parameters

**characterSet**
Type: System.String
The characters of the encoding.

**msbComesFirst**
Type: System.Boolean
**true** to begin with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.

**additionalDecodeCharacters**
Type: System.Collections.Generic.IDictionaryChar, Int32
A dictionary of alias characters, or **null** if no aliases are desired.

**decodeFilterCallback**
Type: CryptSharp.Utility.BaseEncodingDecodeFilterCallback
A callback to map arbitrary characters onto the characters that can be decoded.
See Also

Reference

BaseEncoding Class
BaseEncoding Overload
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The **BaseEncoding** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **GetByteCount** | When overridden in a derived class, calculates the number of bytes produced by encoding a set of characters from the specified character array.  
(Overrides EncodingGetByteCount(Char, Int32, Int32).) |
| **GetBytes(Char, Int32, Int32, Byte, Int32)** | When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array.  
(Overrides EncodingGetBytes(Char, Int32, Int32, Byte, Int32).) |
| **GetBytes(Char, Int32, Int32, Byte, Int32, Int32)** | Converts characters from their text representation to a binary representation. |
| **GetChar** | Gets the character corresponding to the specified value. |
| **GetCharCount** | When overridden in a derived class, calculates the number of characters produced by decoding a sequence of bytes from the specified byte array.  
(Overrides EncodingGetCharCount(Byte, Int32, Int32).) |
| **GetChars(Byte, Int32, Int32, Char, Int32)** | When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array.  
(Overrides EncodingGetChars(Byte, Int32, Int32, Char, Int32).) |
| **GetChars(Byte, Int32, Int32, Char, Int32, Int32)** | Converts bytes from their binary representation to a text representation. |
| **GetMaxByteCount** | When overridden in a derived class, calculates the maximum number of bytes produced by encoding the specified number of characters.  
(Overrides EncodingGetMaxByteCount(Int32).) |
| **GetMaxCharCount** | When overridden in a derived class, calculates the maximum number of characters produced by decoding the specified number of bytes.  
(Overrides EncodingGetMaxCharCount(Int32).) |
| **GetValue** | Gets the value corresponding to the specified character. |
**BaseEncoding.GetByteCount Method**

When overridden in a derived class, calculates the number of bytes produced by encoding a set of characters from the specified character array.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override int GetByteCount(char[] chars, int index, int count)
```

#### Parameters

- **chars**  
  Type: `SystemChar`  
  The character array containing the set of characters to encode.

- **index**  
  Type: `SystemInt32`  
  The index of the first character to encode.

- **count**  
  Type: `SystemInt32`  
  The number of characters to encode.

#### Return Value

Type: `Int32`  
The number of bytes produced by encoding the specified characters.

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td><code>chars</code> is null.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td><code>index</code> or <code>count</code> is less than zero.-or- <code>index</code> and <code>count</code> do not denote a valid range in <code>chars</code>.</td>
</tr>
<tr>
<td>EncoderFallbackException</td>
<td>A fallback occurred (see Understanding Encodings for complete</td>
</tr>
</tbody>
</table>
See Also

Reference

BaseEncoding Class
CryptSharp.Utility Namespace
BaseEncoding.GetBytes Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBytes(Char, Int32, Int32, Byte, Int32)</td>
<td>When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array. (Overrides Encoding.GetBytes(Char, Int32, Int32, Byte, Int32).)</td>
</tr>
<tr>
<td>GetBytes(Char, Int32, Int32, Byte, Int32, Int32)</td>
<td>Converts characters from their text representation to a binary representation.</td>
</tr>
</tbody>
</table>

See Also

Reference

BaseEncoding Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BaseEncoding.GetBytes Method (Char, Int32, Int32, Byte, Int32)

When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public override int GetBytes(
    char[] chars,
    int charIndex,
    int charCount,
    byte[] bytes,
    int byteIndex
)
```

Parameters

cars
 Type: SystemChar
 The character array containing the set of characters to encode.

charIndex
 Type: SystemInt32
 The index of the first character to encode.

charCount
 Type: SystemInt32
 The number of characters to encode.

bytes
 Type: SystemByte
 The byte array to contain the resulting sequence of bytes.

byteIndex
 Type: SystemInt32
 The index at which to start writing the resulting sequence of bytes.

Return Value
 Type: Int32
The actual number of bytes written into bytes.

## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>chars is null.-or- bytes is null.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>charIndex or charCount or byteIndex is less than zero.-or-</td>
</tr>
<tr>
<td></td>
<td>charIndex and charCount do not denote a valid range in chars.-or-</td>
</tr>
<tr>
<td></td>
<td>byteIndex is not a valid index in bytes.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>bytes does not have enough capacity from byteIndex to the end of the array to accommodate the resulting bytes.</td>
</tr>
<tr>
<td>EncoderFallbackException</td>
<td>A fallback occurred (see Understanding Encodings for complete explanation)-and-EncoderFallback is set to EncoderExceptionFallback.</td>
</tr>
</tbody>
</table>

## See Also

Reference

- **BaseEncoding Class**
- **GetBytes Overload**
- **CryptSharp.Utility Namespace**

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BaseEncodinggetBytes Method (Char, Int32, Int32, Byte, Int32, Int32)

Converts characters from their text representation to a binary representation.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

### C#

```csharp
public int GetBytes(
    char[] chars,
    int charIndex,
    int charCount,
    byte[] bytes,
    int byteIndex,
    int byteCount
)
```

### Parameters

- **chars**
  - Type: SystemChar
  - An input array of characters.

- **charIndex**
  - Type: SystemInt32
  - The index of the first character.

- **charCount**
  - Type: SystemInt32
  - The number of characters to read.

- **bytes**
  - Type: SystemByte
  - An output array of bytes.

- **byteIndex**
  - Type: SystemInt32
  - The index of the first byte.

- **byteCount**
  - Type: SystemInt32
The number of bytes to write.

Return Value
Type: Int32
The number of bytes written.

See Also

Reference
BaseEncoding Class
GetBytes Overload
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**BaseEncoding.GetChar Method**

Gets the character corresponding to the specified value.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public virtual char GetChar(int value)
```

### Parameters

**value**
- Type: System.Int32
- A value.

### Return Value

**Type:** Char
- A character.

### See Also

**Reference**
- BaseEncoding Class
- CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**BaseEncodingGetCharCount Method**

When overridden in a derived class, calculates the number of characters produced by decoding a sequence of bytes from the specified byte array.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override int GetCharCount(
    byte[] bytes,
    int index,
    int count
)
```

### Parameters

- **bytes**
  - Type: `SystemByte`
  - The byte array containing the sequence of bytes to decode.

- **index**
  - Type: `SystemInt32`
  - The index of the first byte to decode.

- **count**
  - Type: `SystemInt32`
  - The number of bytes to decode.

### Return Value

- **Type:** `Int32`
- The number of characters produced by decoding the specified sequence of bytes.

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>bytes is null.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>index or count is less than zero.-or- index and count do not denote a valid range in bytes.</td>
</tr>
<tr>
<td>DecoderFallbackException</td>
<td>A fallback occurred (see Understanding Encodings for complete explanation) and <strong>DecoderFallback</strong> is set to <strong>DecoderExceptionFallback</strong>.</td>
</tr>
</tbody>
</table>

See Also

Reference

**BaseEncoding Class**

**CryptSharp.Utility Namespace**

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
## BaseEncoding.GetChars Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetChars(Byte, Int32, Int32, Char, Int32)</td>
<td>When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array. (Overrides Encoding.GetChars(Byte, Int32, Int32, Char, Int32).)</td>
</tr>
<tr>
<td>GetChars(Byte, Int32, Int32, Char, Int32, Int32)</td>
<td>Converts bytes from their binary representation to a text representation.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- BaseEncoding Class
- CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
BaseEncodingGetChars Method (Byte, Int32, Int32, Char, Int32)

When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public override int GetChars(
    byte[] bytes,
    int byteIndex,
    int byteCount,
    char[] chars,
    int charIndex
)
```

### Parameters

- **bytes**
  - Type: SystemByte
  - The byte array containing the sequence of bytes to decode.

- **byteIndex**
  - Type: SystemInt32
  - The index of the first byte to decode.

- **byteCount**
  - Type: SystemInt32
  - The number of bytes to decode.

- **chars**
  - Type: SystemChar
  - The character array to contain the resulting set of characters.

- **charIndex**
  - Type: SystemInt32
  - The index at which to start writing the resulting set of characters.

### Return Value

- Type: Int32
The actual number of characters written into `chars`.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td><code>bytes</code> is null.-or- <code>chars</code> is null.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td><code>byteIndex</code> or <code>byteCount</code> or <code>charIndex</code> is less than zero.-or- <code>byteIndex</code> and <code>byteCount</code> do not denote a valid range in <code>bytes</code>.-or- <code>charIndex</code> is not a valid index in <code>chars</code>.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td><code>chars</code> does not have enough capacity from <code>charIndex</code> to the end of the array to accommodate the resulting characters.</td>
</tr>
<tr>
<td>DecoderFallbackException</td>
<td>A fallback occurred (see Understanding Encodings for complete explanation)-and-<code>DecoderFallback</code> is set to <code>DecoderExceptionFallback</code>.</td>
</tr>
</tbody>
</table>

**See Also**

- **Reference**
  - BaseEncoding Class
  - GetChars Overload
  - CryptSharp.Utility Namespace

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
BaseEncodingGetChars Method (Byte, Int32, Int32, Char, Int32, Int32)

Converts bytes from their binary representation to a text representation.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public int GetChars(
    byte[] bytes,
    int byteIndex,
    int byteCount,
    char[] chars,
    int charIndex,
    int charCount
)
```

### Parameters

**bytes**
- Type: System.Byte
- An input array of bytes.

**byteIndex**
- Type: System.Int32
- The index of the first byte.

**byteCount**
- Type: System.Int32
- The number of bytes to read.

**chars**
- Type: System.Char
- An output array of characters.

**charIndex**
- Type: System.Int32
- The index of the first character.

**charCount**
- Type: System.Int32
The number of characters to write.

Return Value
Type: Int32
The number of characters written.

See Also

Reference
BaseEncoding Class
GetChars Overload
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BaseEncoding.GetMaxByteCount Method

When overridden in a derived class, calculates the maximum number of bytes produced by encoding the specified number of characters.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public override int GetMaxByteCount(
    int charCount
)
```

Parameters

charCount
Type: System.Int32
The number of characters to encode.

Return Value

Type: Int32
The maximum number of bytes produced by encoding the specified number of characters.

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>charCount is less than zero.</td>
</tr>
<tr>
<td>EncoderFallbackException</td>
<td>A fallback occurred (see Understanding Encodings for complete explanation)-and-EncoderFallback is set to EncoderExceptionFallback.</td>
</tr>
</tbody>
</table>

See Also

Reference

BaseEncoding Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**BaseEncoding GetMaxCharCount Method**  
CryptSharp

When overridden in a derived class, calculates the maximum number of characters produced by decoding the specified number of bytes.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public override int GetMaxCharCount(
    int byteCount
)
```

**Parameters**  
**byteCount**  
Type: System.Int32  
The number of bytes to decode.

**Return Value**  
Type: Int32  
The maximum number of characters produced by decoding the specified number of bytes.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>byteCount is less than zero.</td>
</tr>
<tr>
<td>DecoderFallbackException</td>
<td>A fallback occurred (see Understanding Encodings for complete explanation)-and-DecoderFallback is set to DecoderExceptionFallback.</td>
</tr>
</tbody>
</table>

**See Also**

Reference  
BaseEncoding Class  
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**BaseEncodingGetValue Method**

Gets the value corresponding to the specified character.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public virtual int GetValue(
    char character
)
```

**Parameters**

- `character`
  
  Type: `SystemChar`  
  A character.

**Return Value**

Type: `Int32`  
A value, or `-1` if the character is not part of the encoding.

**See Also**

**Reference**

- BaseEncoding Class  
- CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
The **BaseEncoding** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitMask</td>
<td>The bit mask for a single character in the current encoding.</td>
</tr>
<tr>
<td>BitsPerCharacter</td>
<td>The number of bits per character in the current encoding.</td>
</tr>
<tr>
<td>MsbComesFirst</td>
<td><em>true</em> if the encoding begins with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.</td>
</tr>
</tbody>
</table>

## See Also

Reference

**BaseEncoding Class**

**CryptSharp.Utility Namespace**

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
**BaseEncodingBitMask Property**

The bit mask for a single character in the current encoding.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public int BitMask { get; }
```

**Property Value**  
**Type:** Int32

### See Also

**Reference**  
- BaseEncoding Class  
- CryptSharp.Utility Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
BaseEncodingBitsPerCharacter Property

The number of bits per character in the current encoding.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public int BitsPerCharacter { get; }
```

**Property Value**

**Type:** Int32

**See Also**

Reference

BaseEncoding Class  
CryptSharp.Utility Namespace

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**BaseEncodingMsbComesFirst Property**

*true* if the encoding begins with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public bool MsbComesFirst { get; }
```

**Property Value**

Type: **Boolean**

**See Also**

**Reference**

- BaseEncoding Class
- CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**BaseEncodingDecodeFilterCallback Delegate**

A callback to map arbitrary characters onto the characters that can be decoded.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public delegate char BaseEncodingDecodeFilterCallback(  
    char originalCharacter
)
```

**Parameters**

- `originalCharacter`  
  Type: `System.Char`  
  The original character.

**Return Value**

Type: `Char`  
the replacement character.

**See Also**

**Reference**

CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BlowfishCipher Class

Performs low-level encryption and decryption using the Blowfish cipher.

Inheritance Hierarchy

SystemObject  CryptSharp.UtilityBlowfishCipher

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class BlowfishCipher : IDisposable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The **BlowfishCipher** type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCrypt</td>
<td>Uses the cipher to generate a BCrypt hash.</td>
</tr>
<tr>
<td>BCrypt(Byte, Byte, Int32)</td>
<td>Uses the given key, salt, and cost to generate a BCrypt hash.</td>
</tr>
<tr>
<td>BCrypt(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)</td>
<td>Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.</td>
</tr>
<tr>
<td>Create</td>
<td>Creates a Blowfish cipher using the provided key.</td>
</tr>
<tr>
<td>CreateEks(Byte, Byte, Int32)</td>
<td>Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.</td>
</tr>
<tr>
<td>CreateEks(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)</td>
<td>Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.</td>
</tr>
<tr>
<td>Decipher(Byte, Int32)</td>
<td>Reverses the encipherment of eight bytes of data in-place.</td>
</tr>
<tr>
<td>Decipher(UInt32, UInt32)</td>
<td>Reverses the encipherment of eight bytes of data.</td>
</tr>
<tr>
<td>Decipher(Byte, Int32, Byte, Int32)</td>
<td>Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Clears all memory used by the cipher.</td>
</tr>
</tbody>
</table>
**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCryptLength</td>
<td>The number of bytes returned by BCrypt.</td>
</tr>
<tr>
<td>BCryptMagic</td>
<td>The special string used encrypted in the BCrypt algorithm.</td>
</tr>
<tr>
<td>IsKeyWeak</td>
<td>A Blowfish key is weak if one of its S-boxes has a duplicate entry. See <a href="http://www.schneier.com/paper-blowfish-oneyear.html">http://www.schneier.com/paper-blowfish-oneyear.html</a> for more information.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
# BlowfishCipher Methods

The `BlowfishCipher` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCrypt</td>
<td>Uses the cipher to generate a BCrypt hash.</td>
</tr>
<tr>
<td>BCrypt(Byte, Byte, Int32)</td>
<td>Uses the given key, salt, and cost to generate a BCrypt hash.</td>
</tr>
<tr>
<td>BCrypt(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)</td>
<td>Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.</td>
</tr>
<tr>
<td>Create</td>
<td>Creates a Blowfish cipher using the provided key.</td>
</tr>
<tr>
<td>CreateEks(Byte, Byte, Int32)</td>
<td>Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.</td>
</tr>
<tr>
<td>CreateEks(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)</td>
<td>Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.</td>
</tr>
<tr>
<td>Decipher(Byte, Int32)</td>
<td>Reverses the encipherment of eight bytes of data in-place.</td>
</tr>
<tr>
<td>Decipher(UInt32, UInt32)</td>
<td>Reverses the encipherment of eight bytes of data.</td>
</tr>
<tr>
<td>Decipher(Byte, Int32, Byte, Int32)</td>
<td>Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Clears all memory used by the cipher.</td>
</tr>
<tr>
<td>Encipher(Byte, Int32)</td>
<td>Enciphers eight bytes of data in-place.</td>
</tr>
<tr>
<td>Encipher(UInt32, UInt32)</td>
<td>Enciphers eight bytes of data.</td>
</tr>
<tr>
<td>Encipher(Byte, Int32, Byte, Int32)</td>
<td>Enciphers eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
</tbody>
</table>

## See Also

Reference

- `BlowfishCipher Class`
- `CryptSharp.Utility Namespace`

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BlowfishCipher BCrypt Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCrypt</td>
<td>Uses the cipher to generate a BCrypt hash.</td>
</tr>
<tr>
<td>BCrypt(Byte, Byte, Int32)</td>
<td>Uses the given key, salt, and cost to generate a BCrypt hash.</td>
</tr>
<tr>
<td>BCrypt(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)</td>
<td>Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.</td>
</tr>
</tbody>
</table>

See Also

Reference

BlowfishCipher Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BlowfishCipher::BCrypt Method

Uses the cipher to generate a BCrypt hash.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```csharp
public byte[] BCrypt()
```

### Return Value

**Type:** `Byte`  
A BCrypt hash.

### See Also

**Reference**

- BlowfishCipher Class
- BCrypt Overload
- CryptSharp.Utility Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
BlowfishCipherBCrypt Method (Byte, Byte, Int32)

Uses the given key, salt, and cost to generate a BCrypt hash.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public static byte[] BCrypt(
    byte[] key,
    byte[] salt,
    int cost
)
```

### Parameters

- **key**
  - Type: SystemByte  
  - The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter, this method does NOT automatically add a null byte to the key.

- **salt**
  - Type: SystemByte  
  - The salt. This must be 16 bytes.

- **cost**
  - Type: SystemInt32  
  - The expansion cost. This is a value between 4 and 31, specifying the logarithm of the number of iterations.

### Return Value

- Type: Byte  
  - A BCrypt hash.

### See Also

- **Reference**
  - BlowfishCipher Class
  - BCrypt Overload
BlowfishCipherBCrypt Method (Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)  CryptSharp

Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| public static byte[] BCrypt(  
  byte[] key,  
  byte[] salt,  
  int cost,  
  EksBlowfishKeyExpansionFlags flags  
) |

## Parameters

- **key**
  - Type: SystemByte
  - The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter, this method does NOT automatically add a null byte to the key.

- **salt**
  - Type: SystemByte
  - The salt. This must be 16 bytes.

- **cost**
  - Type: SystemInt32
  - The expansion cost. This is a value between 4 and 31, specifying the logarithm of the number of iterations.

- **flags**
  - Type: CryptSharp.UtilityEksBlowfishKeyExpansionFlags
  - Flags modifying the key expansion.

## Return Value

- Type: Byte
  - A BCrypt hash.
See Also

Reference

BlowfishCipher Class
BCrypt Overload
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BlowfishCipherCreate Method

Creates a Blowfish cipher using the provided key.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public static BlowfishCipher Create(byte[] key)
```

### Parameters

**key**

Type: `System.Byte`  
The Blowfish key. This must be between 4 and 56 bytes.

### Return Value

Type: `BlowfishCipher`  
A Blowfish cipher.

## See Also

**Reference**

BlowfishCipher Class  
CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
### BlowfishCipherCreateEks Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateEks(Byte, Byte, Int32)</td>
<td>Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.</td>
</tr>
<tr>
<td>CreateEks(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)</td>
<td>Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.</td>
</tr>
</tbody>
</table>

#### See Also

Reference

**BlowfishCipher Class**

**CryptSharp.Utility Namespace**

---

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
BlowfishCipherCreateEks Method (Byte, Byte, Int32)

Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static BlowfishCipher CreateEks(
    byte[] key,
    byte[] salt,
    int cost
)
```

Parameters

key
Type: System.Byte
The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter, this method does NOT automatically add a null byte to the key.

salt
Type: System.Byte
The salt. This must be 16 bytes.

cost
Type: System.Int32
The expansion cost. This is a value between 4 and 31, specifying the logarithm of the number of iterations.

Return Value
Type: BlowfishCipher
A Blowfish cipher.

See Also

Reference
BlowfishCipher Class
CreateEks Overload
BlowfishCipherCreateEks Method (Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)

Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static BlowfishCipher CreateEks(
    byte[] key,
    byte[] salt,
    int cost,
    EksBlowfishKeyExpansionFlags flags
)
```

**Parameters**

**key**
- Type: **SystemByte**
  - The key. This must be between 1 and 72 bytes. Unlike **BlowfishCrypter**, this method does NOT automatically add a null byte to the key.

**salt**
- Type: **SystemByte**
  - The salt. This must be 16 bytes.

**cost**
- Type: **SystemInt32**
  - The expansion cost. This is a value between 4 and 31, specifying the logarithm of the number of iterations.

**flags**
- Type: **CryptSharp.UtilityEksBlowfishKeyExpansionFlags**
  - Flags modifying the key expansion.

**Return Value**
Type: BlowfishCipher
A Blowfish cipher.

See Also

Reference
BlowfishCipher Class
CreateEks Overload
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
# BlowfishCipherDecipher Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decipher(Byte, Int32)</td>
<td>Reverses the encipherment of eight bytes of data in-place.</td>
</tr>
<tr>
<td>Decipher(UInt32, UInt32)</td>
<td>Reverses the encipherment of eight bytes of data.</td>
</tr>
<tr>
<td>Decipher(Byte, Int32, Byte, Int32)</td>
<td>Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
</tbody>
</table>

## Top

## See Also

### Reference

BlowfishCipher Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
BlowfishCipherDecipher Method (Byte, Int32)

Reverses the encipherment of eight bytes of data in-place.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public void Decipher(
    byte[] buffer,
    int offset
)
```

**Parameters**

- **buffer**
  - Type: System.Byte
  - The buffer containing the data.

- **offset**
  - Type: System.Int32
  - The offset of the first byte to decipher.

**See Also**

- Reference
  - BlowfishCipher Class
  - Decipher Overload
  - CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
BlowfishCipherDecipher Method (UInt32, UInt32)

Reverses the encipherment of eight bytes of data.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```
public void Decipher(
    ref uint xl,
    ref uint xr
)
```

### Parameters

- **xl**
  - Type: System.UInt32
  - The first four bytes.

- **xr**
  - Type: System.UInt32
  - The last four bytes.

### See Also

- **Reference**
  - BlowfishCipher Class
  - Decipher Overload
  - CryptSharp.Utility Namespace

---

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
BlowfishCipherDecipher Method (Byte, Int32, Byte, Int32)

Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```
public void Decipher(
    byte[] inputBuffer,
    int inputOffset,
    byte[] outputBuffer,
    int outputOffset
)
```

**Parameters**

- **inputBuffer**  
  Type: System.Byte  
  The buffer to read enciphered data from.

- **inputOffset**  
  Type: System.Int32  
  The offset of the first enciphered byte.

- **outputBuffer**  
  Type: System.Byte  
  The buffer to write plaintext data to.

- **outputOffset**  
  Type: System.Int32  
  The offset at which to place the first plaintext byte.

**See Also**

**Reference**

- BlowfishCipher Class
- Decipher Overload
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**BlowfishCipherDispose Method**

Clears all memory used by the cipher.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public void Dispose()
```

**Implements**

IDisposableDispose

### See Also

**Reference**

BlowfishCipher Class  
CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**BlowfishCipherEncipher Method**

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encipher(Byte, Int32)</td>
<td>Enciphers eight bytes of data in-place.</td>
</tr>
<tr>
<td>Encipher(UInt32, UInt32)</td>
<td>Enciphers eight bytes of data.</td>
</tr>
<tr>
<td>Encipher(Byte, Int32, Byte, Int32)</td>
<td>Enciphers eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
BlowfishCipherEncipher Method (Byte, Int32)

Enciphers eight bytes of data in-place.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public void Encipher(
    byte[] buffer,
    int offset
)
```

### Parameters

buffer
Type: System.Byte  
The buffer containing the data.

offset
Type: System.Int32  
The offset of the first byte to encipher.

### See Also

Reference

BlowfishCipher Class  
Encipher Overload  
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**BlowfishCipherEncipher Method (UInt32, UInt32)**

Enciphers eight bytes of data.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public void Encipher(
    ref uint xl,
    ref uint xr
)
```

**Parameters**

xl  
Type: System.UInt32  
The first four bytes.

xr  
Type: System.UInt32  
The last four bytes.

**See Also**

Reference  
BlowfishCipher Class  
Encipher Overload  
CryptSharp.Utility Namespace

**Copyright 2010-2014 James F. Bellinger**  
Send comments on this topic to jfb@zer7.com
**BlowfishCipherEncipher Method (Byte, Int32, Byte, Int32)**

Enciphers eight bytes of data from one buffer and places the result in another buffer.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public void Encipher(
    byte[] inputBuffer,  
    int inputOffset,    
    byte[] outputBuffer,  
    int outputOffset
)
```

### Parameters

**inputBuffer**
- Type: System.Byte
- The buffer to read plaintext data from.

**inputOffset**
- Type: System.Int32
- The offset of the first plaintext byte.

**outputBuffer**
- Type: System.Byte
- The buffer to write enciphered data to.

**outputOffset**
- Type: System.Int32
- The offset at which to place the first enciphered byte.

### See Also

**Reference**

BlowfishCipher Class  
Encipher Overload
The **BlowfishCipher** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCryptLength</td>
<td>The number of bytes returned by <code>BCrypt</code>.</td>
</tr>
<tr>
<td>BCryptMagic</td>
<td>The special string used encrypted in the BCrypt algorithm.</td>
</tr>
<tr>
<td>IsKeyWeak</td>
<td>A Blowfish key is weak if one of its S-boxes has a duplicate entry. See <a href="http://www.schneier.com/paper-blowfish-oneyear.html">http://www.schneier.com/paper-blowfish-oneyear.html</a> for more information.</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- BlowfishCipher Class
- CryptSharp.Utility Namespace

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to jfb@zer7.com
**BlowfishCipherBCryptLength Property**

The number of bytes returned by **BCrypt**.

**Namespace:** CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static int BCryptLength { get; }
```

**Property Value**

**Type:** Int32

---

**See Also**

**Reference**

BlowfishCipher Class

CryptSharp.Utility Namespace

---

**Copyright 2010-2014 James F. Bellinger**

Send comments on this topic to **jfb@zer7.com**
**BlowfishCipherBCryptMagic Property**

The special string used encrypted in the BCrypt algorithm.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static string BCryptMagic { get; }
```

**Property Value**

Type: String

**See Also**

Reference

BlowfishCipher Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
A Blowfish key is weak if one of its S-boxes has a duplicate entry. See http://www.schneier.com/paper-blowfish-oneyear.html for more information.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public bool IsKeyWeak { get; }
```

Property Value
Type: Boolean

See Also

Reference
BlowfishCipher Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**DesCipher Class**

Performs low-level encryption and decryption using the DES cipher.

**Inheritance Hierarchy**

```
SystemObject  CryptSharp.UtilityDesCipher
```

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll)  
**Version:** 2.1.0.0

**Syntax**

```csharp
public class DesCipher : IDisposable
```

The **DesCipher** type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Creates a DES cipher using the provided key.</td>
</tr>
<tr>
<td>Crypt</td>
<td>Crypts eight bytes of data in-place.</td>
</tr>
<tr>
<td>Decipher</td>
<td>Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Clears all memory used by the cipher.</td>
</tr>
<tr>
<td>Encipher</td>
<td>Enciphers eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

Cryp**: CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
DesCipher Methods

The DesCipher type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Creates a DES cipher using the provided key.</td>
</tr>
<tr>
<td>Crypt</td>
<td>Crypts eight bytes of data in-place.</td>
</tr>
<tr>
<td>Decipher</td>
<td>Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Clears all memory used by the cipher.</td>
</tr>
<tr>
<td>Encipher</td>
<td>Enciphers eight bytes of data from one buffer and places the result in another buffer.</td>
</tr>
</tbody>
</table>

See Also

Reference

DesCipher Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**DesCipherCreate Method**

Creates a DES cipher using the provided key.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static DesCipher Create(
    byte[] key
)
```

**Parameters**

**key**

Type: System.Byte  
The DES key. This must be eight bytes.

**Return Value**

Type: DesCipher  
A DES cipher.

### See Also

**Reference**

DesCipher Class  
CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
DesCipherCrypt Method

Crypts eight bytes of data in-place.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public void Crypt(
    byte[] buffer,
    int offset,
    int iterations,
    int salt
)
```

### Parameters

**buffer**
- Type: `SystemByte`  
  The buffer to crypt. For traditional DES crypt, this is zero-initialized.

**offset**
- Type: `SystemInt32`  
  The offset into the buffer.

**iterations**
- Type: `SystemInt32`  
  The number of iterations to run.

**salt**
- Type: `SystemInt32`  
  The salt, up to 24 bits.

### See Also

**Reference**
- DesCipher Class
- CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
DesCipherDecipher Method

Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public void Decipher(
    byte[] inputBuffer,
    int inputOffset,
    byte[] outputBuffer,
    int outputOffset
)
```

### Parameters

- **inputBuffer**  
  Type: System.Byte  
  The buffer to read enciphered data from.

- **inputOffset**  
  Type: System.Int32  
  The offset of the first enciphered byte.

- **outputBuffer**  
  Type: System.Byte  
  The buffer to write plaintext data to.

- **outputOffset**  
  Type: System.Int32  
  The offset at which to place the first plaintext byte.

### See Also

**Reference**

- DesCipher Class  
- CryptSharp.Utility Namespace
DesCipherDispose Method

Clears all memory used by the cipher.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```
public void Dispose()
```

Implements

IDisposable

## See Also

Reference

DesCipher Class  
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**DesCipherEncipher Method**

Enciphers eight bytes of data from one buffer and places the result in another buffer.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public void Encipher(  
    byte[] inputBuffer,  
    int inputOffset,  
    byte[] outputBuffer,  
    int outputOffset
)
```

**Parameters**

- **inputBuffer**  
  Type: System.Byte  
  The buffer to read plaintext data from.

- **inputOffset**  
  Type: System.Int32  
  The offset of the first plaintext byte.

- **outputBuffer**  
  Type: System.Byte  
  The buffer to write enciphered data to.

- **outputOffset**  
  Type: System.Int32  
  The offset at which to place the first enciphered byte.

**See Also**

- DesCipher Class
- CryptSharp.Utility Namespace
EksBlowfishKeyExpansionFlags

Enumeration

Modifiers for Expensive Key Schedule (EKS) Blowfish key expansion behavior.

**Namespace:** CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

[C#] [VB] [C++] [F#]

```csharp
[FlagsAttribute]
public enum EksBlowfishKeyExpansionFlags
```

## Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>Default behavior.</td>
</tr>
</tbody>
</table>
| EmulateCryptBlowfishSignExtensionBug           | 1     | The EksBlowfish code in CryptSharp was implemented as per the specification at http://static.usenix.org/event/usenix99/provos/provos_html/node4.html. Many other BCrypt implementations, however, originating with the crypt_blowfish C implementation crypt_blowfish had a sign extension bug that caused up to three characters previous to any 8-bit character to match 0xFF. However, for those who need backwards compatibility for old password databases created with one of these libraries, I have added *support* for the bug this flag.

## See Also

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
Pbkdf2 Class

Implements the PBKDF2 key derivation function.

Inheritance Hierarchy


Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public class Pbkdf2 : Stream
```

The Pbkdf2 type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pbkdf2</td>
<td>Creates a new PBKDF2 stream.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the stream, clearing memory and disposing of the HMAC algorithm. (Overrides Stream.Close.)</td>
</tr>
<tr>
<td>ComputeDerivedKey</td>
<td>Computes a derived key.</td>
</tr>
<tr>
<td>Read(Int32)</td>
<td>Reads from the derived key stream.</td>
</tr>
<tr>
<td>Read(Byte, Int32, Int32)</td>
<td>When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read. (Overrides Stream.ReadByte(Byte, Int32, Int32).)</td>
</tr>
<tr>
<td>Seek</td>
<td>When overridden in a derived class, sets the position within the</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>The maximum number of bytes that can be derived is $2^{32}-1$ times the HMAC size. (Overrides StreamLength.)</td>
</tr>
<tr>
<td>Position</td>
<td>The position within the derived key stream. (Overrides StreamPosition.)</td>
</tr>
</tbody>
</table>

## Examples

### Computing a Derived Key

```csharp
using CryptSharp.Utility;

// Compute a 128-byte derived key using HMAC-SHA256, byte[] derivedKey = Pbkdf2.ComputeDerivedKey(new HMACSHA256(key), salt, iterations);
```

### Creating a Derived Key Stream

```csharp
using System.IO;
using CryptSharp.Utility;

// Create a stream using HMAC-SHA512, 1000 iteration
Stream derivedKeyStream = new Pbkdf2(new HMACSHA512(key), salt, iterations);
```

## See Also

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**Pbkdf2 Constructor**

Creates a new PBKDF2 stream.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public Pbkdf2(
    KeyedHashAlgorithm hmacAlgorithm,
    byte[] salt,
    int iterations
)
```

**Parameters**

- **hmacAlgorithm**
  - The HMAC algorithm to use, for example `HMACSHA256`. Make sure to set `Key`.

- **salt**
  - Type: `System.Byte`  
  - The salt. A unique salt means a unique PBKDF2 stream, even if the original key is identical.

- **iterations**
  - Type: `System.Int32`  
  - The number of iterations to apply.

### See Also

**Reference**

- Pbkdf2 Class
- CryptSharp.Utility Namespace

---

_Copyright 2010-2014 James F. Bellinger_  
_Send comments on this topic to jfb@zer7.com_
Pbkdf2 Methods

The Pbkdf2 type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the stream, clearing memory and disposing of the HMAC algorithm.</td>
</tr>
<tr>
<td></td>
<td>(overrides Stream.Close.)</td>
</tr>
<tr>
<td>ComputeDerivedKey</td>
<td>Computes a derived key.</td>
</tr>
<tr>
<td>Read(Int32)</td>
<td>Reads from the derived key stream.</td>
</tr>
<tr>
<td>Read(Byte, Int32, Int32)</td>
<td>When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read. (overrides Stream.Read(Byte, Int32, Int32).)</td>
</tr>
<tr>
<td>Seek</td>
<td>When overridden in a derived class, sets the position within the current stream. (overrides Stream.Seek(Int64, SeekOrigin).)</td>
</tr>
</tbody>
</table>

See Also

Reference

Pbkdf2 Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**Pbkdf2Close Method**

Closes the stream, clearing memory and disposing of the HMAC algorithm.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```
public override void Close()
```

**See Also**

Reference

Pbkdf2 Class  
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Pbkdf2ComputeDerivedKey Method

Computes a derived key.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static byte[] ComputeDerivedKey(
    KeyedHashAlgorithm hmacAlgorithm,  
    byte[] salt,  
    int iterations,  
    int derivedKeyLength
)
```

**Parameters**

- **hmacAlgorithm**  
  The HMAC algorithm to use, for example HMACSHA256. Make sure to set Key.

- **salt**  
  Type: System.Byte  
  The salt. A unique salt means a unique derived key, even if the original key is identical.

- **iterations**  
  Type: System.Int32  
  The number of iterations to apply.

- **derivedKeyLength**  
  Type: System.Int32  
  The desired length of the derived key.

**Return Value**

Type: Byte  
The derived key.

**See Also**

**Reference**
Pbkdf2 Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Pbkdf2Read Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read(Int32)</td>
<td>Reads from the derived key stream.</td>
</tr>
<tr>
<td>Read(Byte, Int32, Int32)</td>
<td>When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read. (Override Stream.Read(Byte, Int32, Int32).)</td>
</tr>
</tbody>
</table>

See Also

Reference

Pbkdf2 Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Pbkdf2Read Method (Int32)

Reads from the derived key stream.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## Syntax

```csharp
public byte[] Read(int count)
```

### Parameters

**count**
Type: System.Int32  
The number of bytes to read.

### Return Value

Type: Byte  
Bytes from the derived key stream.

## See Also

**Reference**

Pbkdf2 Class  
Read Overload  
CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
Pbkdf2Read Method (Byte, Int32, Int32)  
CryptSharp

When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
<th>Copy</th>
</tr>
</thead>
</table>

```csharp
public override int Read(
    byte[] buffer,
    int offset,
    int count
)
```

**Parameters**

- **buffer**  
  Type: `SystemByte`  
  An array of bytes. When this method returns, the buffer contains the specified byte array with the values between `offset` and `(offset + count - 1)` replaced by the bytes read from the current source.

- **offset**  
  Type: `SystemInt32`  
  The zero-based byte offset in `buffer` at which to begin storing the data read from the current stream.

- **count**  
  Type: `SystemInt32`  
  The maximum number of bytes to be read from the current stream.

**Return Value**

Type: `Int32`  
The total number of bytes read into the buffer. This can be less than the number of bytes requested if that many bytes are not currently available, or zero (0) if the end of the stream has been reached.

### Exceptions
<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>The sum of <code>offset</code> and <code>count</code> is larger than the buffer length.</td>
</tr>
<tr>
<td>ArgumentNullException</td>
<td><code>buffer</code> is null.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td><code>offset</code> or <code>count</code> is negative.</td>
</tr>
<tr>
<td>IOException</td>
<td>An I/O error occurs.</td>
</tr>
<tr>
<td>NotSupportedException</td>
<td>The stream does not support reading.</td>
</tr>
<tr>
<td>ObjectDisposedException</td>
<td>Methods were called after the stream was closed.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

Pbkdf2 Class

Read Overload

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
Pbkdf2Seek Method

When overridden in a derived class, sets the position within the current stream.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

**C#**

```csharp
public override long Seek(
    long offset,
    SeekOrigin origin
)
```

**Parameters**

- **offset**
  - Type: `System.Int64`
  - A byte offset relative to the `origin` parameter.

- **origin**
  - Type: `System.IO.SeekOrigin`
  - A value of type `SeekOrigin` indicating the reference point used to obtain the new position.

**Return Value**

Type: `Int64`

The new position within the current stream.

### Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IOException</code></td>
<td>An I/O error occurs.</td>
</tr>
<tr>
<td><code>NotSupportedException</code></td>
<td>The stream does not support seeking, such as if the stream is constructed from a pipe or console output.</td>
</tr>
<tr>
<td><code>ObjectDisposedException</code></td>
<td>Methods were called after the stream was closed.</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
Pbkdf2 Properties

The Pbkdf2 type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>The maximum number of bytes that can be derived is $2^{32}-1$ times the HMAC size. (Overrides StreamLength.)</td>
</tr>
<tr>
<td>Position</td>
<td>The position within the derived key stream. (Overrides StreamPosition.)</td>
</tr>
</tbody>
</table>

See Also

Reference

Pbkdf2 Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
**Pbkdf2Length Property**

The maximum number of bytes that can be derived is $2^{32} - 1$ times the HMAC size.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override long Length { get; }
```

**Property Value**  
Type: Int64

### See Also

**Reference**  
Pbkdf2 Class  
CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**Pbkdf2Position Property**

The position within the derived key stream.

**Namespace:** CryptSharp.Utily  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public override long Position { get; set; }
```

### Property Value

Type: **Int64**

### See Also

**Reference**

Pbkdf2 Class  
CryptSharp.Utility Namespace

---

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
**Salsa20Core Class**

Implements the Salsa20 hash function.

**Inheritance Hierarchy**

*System.Object* → *CryptSharp.Utility.Salsa20Core*

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```
public static class Salsa20Core
```

The **Salsa20Core** type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="salsa20core.png" alt="Salsa20Core" /></td>
<td>Applies the Salsa20 hash function. It maps a 16 element input to an output of the same size.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

*CryptSharp.Utility Namespace*

*Copyright 2010-2014 James F. Bellinger*  
Send comments on this topic to jfb@zer7.com
Salsa20Core Methods

The Salsa20Core type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute</td>
<td>Applies the Salsa20 hash function. It maps a 16 element input to an output of the same size.</td>
</tr>
</tbody>
</table>

See Also

Reference

Salsa20Core Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
Salsa20CoreCompute Method

Applies the Salsa20 hash function. It maps a 16 element input to an output of the same size.

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

```csharp
public static void Compute(
    int rounds,
    uint[] input,
    int inputOffset,
    uint[] output,
    int outputOffset
)
```

Parameters

rounds
Type: SystemInt32
The number of rounds. SCrypt uses 8.

input
Type: SystemUInt32
The input buffer.

inputOffset
Type: SystemInt32
The offset into the input buffer.

output
Type: SystemUInt32
The output buffer.

outputOffset
Type: SystemInt32
The offset into the output buffer.

See Also
**SCrypt Class**

Implements the SCrypt key derivation function.

**Inheritance Hierarchy**

System\Object CryptSharp.Utility\SCrypt

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>
| **public static class SCrypt**

The **SCrypt** type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="thumb" /> <strong>ComputeDerivedKey</strong></td>
<td>Computes a derived key.</td>
</tr>
<tr>
<td><img src="image" alt="thumb" /> <strong>GetEffectivePbkdf2Salt</strong></td>
<td>The SCrypt algorithm creates a salt which it then uses as a one-iteration PBKDF2 key stream with SHA256 HMAC. This method lets you retrieve this intermediate salt.</td>
</tr>
<tr>
<td><img src="image" alt="thumb" /> <strong>GetStream</strong></td>
<td>Creates a derived key stream from which a derived key can be read.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com
SCrypt Methods

The SCrypt type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComputeDerivedKey</td>
<td>Computes a derived key.</td>
</tr>
<tr>
<td>GetEffectivePbkdf2Salt</td>
<td>The SCrypt algorithm creates a salt which it then uses as a one-iteration</td>
</tr>
<tr>
<td></td>
<td>PBKDF2 key stream with SHA256 HMAC. This method lets you retrieve this intermediate salt.</td>
</tr>
<tr>
<td>GetStream</td>
<td>Creates a derived key stream from which a derived key can be read.</td>
</tr>
</tbody>
</table>

See Also

Reference

SCrypt Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
SCrypt

Syntax

```csharp
public static byte[] ComputeDerivedKey(
    byte[] key,
    byte[] salt,
    int cost,
    int blockSize,
    int parallel,
    Nullable<int> maxThreads,
    int derivedKeyLength
)
```

Parameters

**key**
Type: System.Byte
The key to derive from.

**salt**
Type: System.Byte
The salt. A unique salt means a unique SCrypt stream, even if the original key is identical.

**cost**
Type: System.Int32
The cost parameter, typically a fairly large number such as 262144. Memory usage and CPU time scale approximately linearly with this parameter.

**blockSize**
Type: System.Int32
The mixing block size, typically 8. Memory usage and CPU time scale approximately linearly with this parameter.

**parallel**
Type: System\n
The level of parallelism, typically 1. CPU time scales approximately linearly with this parameter.

$maxThreads$

Type: System\n
The maximum number of threads to spawn to derive the key. This is limited by the parallel value. nu1 will use as many threads as possible.

$derivedKeyLength$

Type: System\n
The desired length of the derived key.

Return Value
Type: Byte

The derived key.

See Also

Reference

SCrypt Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
The SCrypt algorithm creates a salt which it then uses as a one-iteration PBKDF2 key stream with SHA256 HMAC. This method lets you retrieve this intermediate salt.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static byte[] GetEffectivePbkdf2Salt(
    byte[] key,
    byte[] salt,
    int cost,
    int blockSize,
    int parallel,
    Nullable<int> maxThreads
)
```

### Parameters

**key**
- Type: `SystemByte`  
- The key to derive from.

**salt**
- Type: `SystemByte`  
- The salt. A unique salt means a unique SCrypt stream, even if the original key is identical.

**cost**
- Type: `SystemInt32`  
- The cost parameter, typically a fairly large number such as 262144. Memory usage and CPU time scale approximately linearly with this parameter.

**blockSize**
- Type: `SystemInt32`  
- The mixing block size, typically 8. Memory usage and CPU time scale approximately linearly with this parameter.
**parallel**
Type: `SystemInt32`
The level of parallelism, typically 1. CPU time scales approximately linearly with this parameter.

**maxThreads**
Type: `SystemNullableInt32`
The maximum number of threads to spawn to derive the key. This is limited by the `parallel` value. `null` will use as many threads as possible.

**Return Value**
Type: `Byte`
The effective salt.

**See Also**
**Reference**
- SCrypt Class
- CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
SCryptGetStream Method

Creates a derived key stream from which a derived key can be read.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### Syntax

```csharp
public static Pbkdf2 GetStream(
    byte[] key,
    byte[] salt,
    int cost,
    int blockSize,
    int parallel,
    Nullable<int> maxThreads
)
```

### Parameters

- **key**
  - Type: **SystemByte**
  - The key to derive from.

- **salt**
  - Type: **SystemByte**
  - The salt. A unique salt means a unique scrypt stream, even if the original key is identical.

- **cost**
  - Type: **SystemInt32**
  - The cost parameter, typically a fairly large number such as 262144. Memory usage and CPU time scale approximately linearly with this parameter.

- **blockSize**
  - Type: **SystemInt32**
  - The mixing block size, typically 8. Memory usage and CPU time scale approximately linearly with this parameter.

- **parallel**
  - Type: **SystemInt32**
The level of parallelism, typically 1. CPU time scales approximately linearly with this parameter.

$maxThreads$

Type: System.Nullable<Int32>

The maximum number of threads to spawn to derive the key. This is limited by the $parallel$ value. **null** will use as many threads as possible.

Return Value

Type: Pbkdf2

The derived key stream.

See Also

Reference

SCrypt Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com
SecureComparison Class

Provides comparison methods resistant to timing attack.

Inheritance Hierarchy

System
Object  CryptSharp.Utility
SecureComparison

Namespace: CryptSharp.Utility
Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

Syntax

<table>
<thead>
<tr>
<th>C#</th>
<th>VB</th>
<th>C++</th>
<th>F#</th>
</tr>
</thead>
</table>

```
public static class SecureComparison
```

The SecureComparison type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟 Equals</td>
<td></td>
</tr>
</tbody>
</table>

Comparer two strings in a timing-insensitive manner.

See Also

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
SecureComparison Methods

The SecureComparison type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🟢 equals</td>
<td>Compares two strings in a timing-insensitive manner.</td>
</tr>
</tbody>
</table>

See Also

Reference

SecureComparison Class
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger
Send comments on this topic to jfb@zer7.com
SecureComparisonEquals Method

Compares two strings in a timing-insensitive manner.

**Namespace:** CryptSharp.Utility  
**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

**Syntax**

```csharp
public static bool Equals(
    string potentialAttackerSuppliedString,
    string referenceString
)
```

**Parameters**

- `potentialAttackerSuppliedString`  
  Type: System.String  
  The string controlled by a potential attacker.

- `referenceString`  
  Type: System.String  
  The string not controlled by a potential attacker.

**Return Value**

Type: **Boolean**  
**true** if the strings are equal.

**Remarks**

If the reference string is zero-length, this method does not protect it against timing attacks. If the reference string is extremely long, memory caching effects may reveal that fact.

**See Also**

**Reference**

SecureComparison Class  
CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger  
Send comments on this topic to jfb@zer7.com