

```
1 using System;
2 using System.IO;
3 using System.Security.Cryptography;
4 using System.Text;
5
6 using static SmokeScreen.Modules.Logging;
7 using static SmokeScreen.Modules.Common;
8
9 namespace SmokeScreen.Modules
10 {
11     public static class Cryptography
12     {
13         //Reference: Microsoft Docs System.Security.Cryptography -> AesCryptoServiceProvider Class
14         public static class AES
15         {
16             public static string Encrypt(string symmetricKey, string message, out string IV)
17             {
18                 using (Aes aes = new AesCryptoServiceProvider())
19                 {
20                     //Accepts 16, 24, or 32 byte Keysize - [Sha 256 hash is 32 bytes]
21                     aes.Key = Convert.FromBase64String(symmetricKey);
22                     IV = Convert.ToBase64String(aes.IV);
23
24                     using (MemoryStream memoryStream = new MemoryStream())
25                     {
26                         using (CryptoStream cryptoStream = new CryptoStream
27                             (memoryStream, aes.CreateEncryptor(),
28                             CryptoStreamMode.Write))
29                         {
30                             byte[] plaintext = encoding.GetBytes(message);
31                             cryptoStream.Write(plaintext, 0, plaintext.Length);
32                         }
33                         message = Convert.ToBase64String(memoryStream.ToArray());
34                     }
35                 }
36                 return message;
37             }
38
39             public static string Decrypt(string symmetricKey, string message,
40                 string IV)
41             {
42                 using (Aes aes = new AesCryptoServiceProvider())
43                 {
44                     aes.Key = Convert.FromBase64String(symmetricKey);
45                     aes.IV = Convert.FromBase64String(IV);
46
47                     using (MemoryStream memoryStream = new MemoryStream())
48                     {
49                         using (CryptoStream cryptoStream = new CryptoStream
```

```

...ce\repos\SmokeScreen2\SmokeScreen\Modules\Cryptography.cs 2
    (memoryStream, aes.CreateDecryptor(),
    CryptoStreamMode.Write))
    {
47         byte[] cipherText = Convert.FromBase64String
48         (message);
49         cryptoStream.Write(cipherText, 0, cipherText.Length);
50     }
51     message = encoding.GetString(memoryStream.ToArray());
52 }
53 }
54 return message;
55 }
56 }
57
58 //Reference: Microsoft Docs System.Security.Cryptography ->
59 RijndaelManaged Class
60 public static class RIJ
61 {
62     public static string Encrypt(string symmetricKey, string message, out
63     string IV)
64     {
65         using (RijndaelManaged rij = new RijndaelManaged())
66         {
67             //Converts to 24 Bytes
68             byte[] symKey = Convert.FromBase64String(symmetricKey);
69             Array.Resize(ref symKey, 8);
70
71             rij.Key = Convert.FromBase64String(symmetricKey);
72             IV = Convert.ToBase64String(rij.IV);
73
74             rij.Padding = PaddingMode.Zeros;
75
76             using (MemoryStream memoryStream = new MemoryStream())
77             {
78                 using (CryptoStream cryptoStream = new CryptoStream
79                 (memoryStream, rij.CreateEncryptor(),
80                 CryptoStreamMode.Write))
81                 {
82                     byte[] plainText = encoding.GetBytes(message);
83                     cryptoStream.Write(plainText, 0, plainText.Length);
84                 }
85                 message = Convert.ToBase64String(memoryStream.ToArray());
86             }
87         }
88         return message;
89     }
90 }

91 public static string Decrypt(string symmetricKey, string message,
92 string IV)
93 {
94     using (RijndaelManaged rij = new RijndaelManaged())
95     {

```

```
91         rij.Key = Convert.FromBase64String(symmetricKey);
92         rij.IV = Convert.FromBase64String(IV);
93
94         rij.Padding = PaddingMode.Zeros;
95
96         using (MemoryStream memoryStream = new MemoryStream())
97         {
98             using (CryptoStream cryptoStream = new CryptoStream
100                 (memoryStream, rij.CreateDecryptor(),
101                 CryptoStreamMode.Write))
102             {
103                 byte[] cipherText = Convert.FromBase64String
104                     (message);
105                 cryptoStream.Write(cipherText, 0, cipherText.Length);
106             }
107             message = encoding.GetString(memoryStream.ToArray());
108         }
109     }
110     return message;
111 }
112
113 //Reference: Microsoft Docs System.Security.Cryptography ->
114 //RC2CryptoServiceProvider Class
115 public static class RC2
116 {
117     public static string Encrypt(string symmetricKey, string message, out
118         string IV)
119     {
120         using (RC2CryptoServiceProvider arc2 = new
121             RC2CryptoServiceProvider())
122         {
123             //Converts to 8 Bytes
124             byte[] symKey = Convert.FromBase64String(symmetricKey);
125             Array.Resize(ref symKey, 8);
126
127             //Accepts 8 byte Keysize
128             arc2.Key = symKey;
129             IV = Convert.ToBase64String(arc2.IV);
130
131             using (MemoryStream memoryStream = new MemoryStream())
132             {
133                 using (CryptoStream cryptoStream = new CryptoStream
134                     (memoryStream, arc2.CreateEncryptor(),
135                     CryptoStreamMode.Write))
136                 {
137                     byte[] plainText = encoding.GetBytes(message);
138                     cryptoStream.Write(plainText, 0, plainText.Length);
139                 }
140                 message = Convert.ToBase64String(memoryStream.ToArray());
141             }
142         }
143     }
144 }
```

```
135         return message;
136     }
137
138     public static string Decrypt(string symmetricKey, string message,
139     string IV)
140     {
141         using (RijndaelManaged rij = new RijndaelManaged())
142         {
143             rij.Key = Convert.FromBase64String(symmetricKey);
144             rij.IV = Convert.FromBase64String(IV);
145
146             using (MemoryStream memoryStream = new MemoryStream())
147             {
148                 using (CryptoStream cryptoStream = new CryptoStream
149                 (memoryStream, rij.CreateDecryptor(), CryptoStreamMode.Read))
150                 {
151                     byte[] cipherText = Convert.FromBase64String
152                     (message);
153                     cryptoStream.Write(cipherText, 0, cipherText.Length);
154                 }
155                 message = encoding.GetString(memoryStream.ToArray());
156             }
157         }
158         return message;
159     }
160
161     public static class TDES
162     {
163         public static string Encrypt(string symmetricKey, string message, out
164         string IV)
165         {
166             using (TripleDESCryptoServiceProvider des = new
167             TripleDESCryptoServiceProvider())
168             {
169                 //Converts to 24 Bytes
170                 byte[] symKey = Convert.FromBase64String(symmetricKey);
171                 Array.Resize(ref symKey, 24);
172
173                 //Accepts 16, 24 byte Keysizes
174                 des.Key = symKey;
175                 IV = Convert.ToBase64String(des.IV);
176
177                 using (MemoryStream memoryStream = new MemoryStream())
178                 {
179                     using (CryptoStream cryptoStream = new CryptoStream
180                     (memoryStream, des.CreateEncryptor(),
181                     CryptoStreamMode.Write))
182                     {
183                         byte[] plainText = encoding.GetBytes(message);
184                         cryptoStream.Write(plainText, 0, plainText.Length);
185                     }
186                 }
187             }
188         }
189     }
190 }
```

```
180         message = Convert.ToBase64String(memoryStream.ToArray());
181     }
182 }
183 return message;
184 }
185
186 public static string Decrypt(string symmetricKey, string message,
187     string IV)
188 {
189     using (TripleDESCryptoServiceProvider des = new
190         TripleDESCryptoServiceProvider())
191     {
192         //Converts to 24 Bytes
193         byte[] symKey = Convert.FromBase64String(symmetricKey);
194         Array.Resize(ref symKey, 24);
195
196         //24 Byte Key
197         des.Key = symKey;
198         des.IV = Convert.FromBase64String(IV);
199
200         using (MemoryStream memoryStream = new MemoryStream())
201         {
202             using (CryptoStream cryptoStream = new CryptoStream
203                 (memoryStream, des.CreateDecryptor(),
204                 CryptoStreamMode.Write))
205             {
206                 byte[] cipherText = Convert.FromBase64String
207                     (message);
208                 cryptoStream.Write(cipherText, 0, cipherText.Length);
209             }
210             message = encoding.GetString(memoryStream.ToArray());
211         }
212     }
213     return message;
214 }
215
216 /// <summary>
217 /// Used https://docs.microsoft.com/en-us/dotnet/api/
218 /// system.security.cryptography.sha256managed for
219 /// Example of How to Implement a Sha256 Hash with C#
220 /// </summary>
221 public static class Sha256Hash
222 {
223     public static string Generate(string password)
224     {
225         if (string.IsNullOrEmpty(password))
226         {
227             return string.Empty;
228         }
229         else
230         {
231             // ...
232         }
233     }
234 }
```

```
226         byte[] hash;
227
228         using (SHA256 generator = SHA256.Create())
229         {
230             hash = generator.ComputeHash(encoding.GetBytes
231                                     (password));
232         }
233         return Convert.ToBase64String(hash);
234     }
235 }
236
237 public static bool Compare(string password1, string password2)
238 {
239     if (string.IsNullOrEmpty(password1) || string.IsNullOrEmpty
240                                     (password2))
241     {
242         return false;
243     }
244     if (password1 == password2)
245     {
246         return true;
247     }
248     else
249     {
250         return false;
251     }
252 }
253
254
255 public enum Algorithm
256 {
257     AES = 0,
258     DES = 1,
259     RIJ = 2
260 }
261
262 }
263 }
```