Ford Security DLL

The purpose of this DLL is to allow approved applications to generate the correct securityKey response to securitySeed requests based upon the applicable Ford security algorithm. In addition, the DLL allows approved applications to decrypt fixed byte information from encrypted strings. The DLL must be unlocked each time with a personalized key which has an embedded expiration date. The DLL can be linked or used directly. This is dependent on whether the calling application is scripted or compiled. The algorithm is built to support both standard programming as well as .Net technology. The two options are detailed bellow.

1. .Net Capable Application:

The algorithm for this option consists of two DLLs, KeyDown.DLL and FordKeyUp.DLL. The KeyDown.DLL is written in standard C++ while FordKeyUp.DLL is written in managed C++/CLI. End users need to have both DLLs in a directory known to the application that uses them. However, only FordKeyUp.dll needs to be linked to the application (if required). No marshaling or interop is required. In addition, msvcr100.dll and msvcp100.dll must also be installed in a directory known to the application that uses them.

1.1. APIs

- bool Povolit(array <unsigned char> ^EncryptedDllKey)
  The purpose of this function is to unlock the security algorithm DLL. It uses an encrypted **16** byte access key. This function must be invoked prior to calling any other DLL functions. If this function is not invoked first, the other functions will return default information. The access keys can be requested through the Ford NetCom group.
  Note that it is only necessary to call this function once during the application execution. This call shall be located in an area of code (e.g., startup) separate from where GetKeyAsInt or GetKeyAsHex are called and shall not be repeated every time prior to calling GetKeyAsInt or GetKeyAsHex.

  Parameter(s):
  EncryptedDllKey: An array of 16 bytes (unsigned char). This is an encrypted access key to unlock the algorithm DLL.

  Return Value:
  true if the DLL is unlocked successfully, false otherwise.

- bool Povolit32(array <unsigned char> ^EncryptedDllKey)
  The purpose of this function is to unlock the security algorithm DLL for specific Fixed Bytes. It uses an encrypted **32** byte access key. This function must be invoked prior to calling any other DLL functions. If this function is not invoked first, the other functions will return default information. The access keys can be requested through the Ford NetCom group.
  Note that it is only necessary to call this function once during the application execution. This call shall be located in an area of code (e.g., startup) separate from where GetKeyAsInt or GetKeyAsHex are called and shall not be repeated every time prior to calling GetKeyAsInt or GetKeyAsHex.

  Parameter(s):
  EncryptedDllKey: An array of 32 bytes (unsigned char). This is an encrypted access key to unlock the algorithm DLL.

  Return Value:
  true if the DLL is unlocked successfully, false otherwise.

- static int GetKeyAsInt(unsigned char S1, unsigned char S2, unsigned char S3, array<unsigned char>^ FixedBytes)
  This function is used to return the securityKey value as a 32 bit integer.

  Parameters:
  S1, S2, S3 are the three random securitySeed bytes returned from the ECU.
  FixedBytes is the array that holds the five fixed challenge bytes.

  Return Value:
  32 bit integer.

- static String^ GetKeyAsHex(unsigned char S1, unsigned char S2, unsigned char S3, array<unsigned char>^ FixedBytes);
  This function is used to return the security key value as a hex string.

  Parameters:
  S1, S2, S3 are the three random securitySeed bytes returned from the ECU.
  FixedBytes is the array that holds the five fixed challenge bytes.

  Return Value:
  A hex string.

- unsigned char Security::GetKey(array<unsigned char>^ SecuritySeed, array<unsigned char>^ FixedBytes, int AlgorithmType, array<unsigned char>^ SecurityKey)

  This function is used to obtain the security key value.

  Parameters:
  SecuritySeed is the 16 byte random securitySeed bytes returned from the ECU.
  FixedBytes is the array that holds the unencrypted challenge bytes.
  AlgorithmType is the Ford algorithm type. 1 = Diagnostic Application Security Algorithm, 2 = Diagnostic Programming Security Algorithm.
  SecurityKey is the byte array that will contain the calculated securityKey. The calling application must allocate 16 bytes for Diagnostic Application Security Algorithm key or 18 bytes for Diagnostic Programming Security Algorithm key.

  Return Value:
  1 : Success and the key is returned in SecurityKey parameter.

- unsigned char Security::DecryptBytesWithFordAlgorithm(array<unsigned char>^ ByteEncrypted, array<unsigned char>^ ByteDecrypted)
  This function is used to decrypt a byte array using one of the Ford decryption algorithms.

  Parameters:
  ByteEncrypted is a 16-byte encrypted input value.
  ByteDecrypted is the decrypted output value.

  Return Value:
  0 : Failure (ByteDecrypted will be all zeros)
  1, 2, 3, 4, 5, 6, 7: This corresponds to the following encryption algorithms (in order): algorithm1, algorithm2, algorithm3, development application, development programming, production application, production programming.

- unsigned char Security::DecryptBytesWithFordAlgorithm(int Alg, array<unsigned char>^ ByteEncrypted, array<unsigned char>^ ByteDecrypted)
  This function is used to decrypt a byte array using one of the Ford decryption algorithms.

  Parameters:
  Alg is the Ford algorithm to decrypt with (1, 2, or 3)
  ByteEncrypted is a 16-byte encrypted input value.
  ByteDecrypted is the decrypted output value.

  Return Value:
  0 : Failure (ByteDecrypted will be all zeros)
  1, 2, 3, 4, 5, 6, 7: This corresponds to the following encryption algorithms (in order): algorithm1, algorithm2, algorithm3, development application, development programming, production application, production programming.

- unsigned char Security::EncryptBytesWithFordAlgorithm(int Alg, array<unsigned char>^ FixedBytes, array<unsigned char>^ ByteEncrypted)

  Parameters:
  Alg is the Ford algorithm to encrypt with (1, 2, or 3)
  FixedBytes is a 16-byte input value to encrypt.
  ByteEncrypted is the encrypted output value.

  Return Value:
  0 : Failure (ByteEncrypted will be all zeros)
  1, 2, 3, 4, 5, 6, 7: This corresponds to the following encryption algorithms (in order): algorithm1, algorithm2, algorithm3, development application, development programming, production application, production programming.

1.2. Examples:

.Net C++/CLI

```
array <unsigned char> ^FixedBytes1 = {0x1a, 0x2b, 0x3c, 0x4d, 0x5e};
array <unsigned char> ^EncryptedData = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

array <unsigned char> ^ SecuritySeed = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88,
0x99, 0xAA, 0xBB, 0xCC};

array <unsigned char> ^ FixedByte2= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

array <unsigned char> ^ SecurityKey = gcnew array<unsigned char>(18);

FordKeyUp::Security^ s = gcnew FordKeyUp::Security;
FordKeyUp::Security::Povolit(EncryptedData);

String ^Text = FordKeyUp::Security::GetKeyAsHex(0x4E, 0x96, 0x8A, FixedBytes1);

String ^Text = FordKeyUp::Security:: GetKey(SecuritySeed, FixedByte2, 2, SecurityKey);
```

VB.Net

```
Imports FordKeyUp
Public Class Form1

    Private Sub btnSecurity_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnSecurity.Click

        Dim key As Integer

        IsModuleLoaded("FordKeyUp")
        Dim EncryptedUserData As Byte() = {&H00, &H00, &H00, &H00, &H00, &H00,
        &H00, &H00, &H00, &H00, &H00, &H00, &H00, &H00, &H00, &H00}
        Dim FixedBytesIn As Byte() = {&H1A, &H2B, &H3C, &H4D, &H5E}

        Dim unlock As Boolean = FordKeyUp.Security.Povolit(EncryptedUserData)
        key = FordKeyUp.Security.GetKeyAsInt(&H4E, &H96, &H8A, FixedBytesIn)

    End Sub

    Private Sub btnDecrypt_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnGetDevKey.Click

        Dim key As Integer
        Dim Result As Byte
        IsModuleLoaded("FordKeyUp")
        Dim ByteEncrypted As Byte() = {&H00, &H00, …, &H00} 'This is a 16 bytes array
        Dim ByteDecrypted As Byte() ' This is the bytes array to hold the decrypted value
        (pass by ref)
        Results=FordKeyUp.Security.DecryptBytesWithFordAlgorithm(ByteEncrypted,
        ByteDecrypted)

    End Sub
```

```
    Function IsModuleLoaded(ByVal moduleName As String) As Boolean
        ' Get the module in the process according to the module name.
        Dim hMod As IntPtr = GetModuleHandle(moduleName)
        Return (hMod <> IntPtr.Zero)
    End Function

    Function GetModuleHandle(ByVal moduleName As String) As IntPtr
    End Function

End Class
```

2. Non .Net Applications:

The algorithm for this option consists of two DLLs, KeyDown.DLL and KeyDownWrapper.DLL. The KeyDown.DLL is written in standard C++ code while KeyDownWrapper.DLL is written in standard C. End users need to have both DLLs in a directory known to the application that uses them. However, only KeyDownWrapper.dll needs to be linked to the application (for certain programming languages).

2.1. APIs

The KeyDownWrapper.DLL exposes the following functions:

- bool Povolit(unsigned char EncryptedDllKey[16])
  The purpose of this function is to unlock the security algorithm DLL. It uses an encrypted 16 byte access key. This function must be invoked prior to requesting the security key. If this function is not invoked first, the security key requested will always be zero. The access keys can be requested through the Ford NetCom group.
  Note that it is only necessary to call this function once during the application execution. This call shall be located in an area of code (e.g., startup) separate from where GetKey is called and shall not be repeated every time prior to calling GetKey.

  Parameter(s):
  EncryptedDllKey: An array of 16 bytes (unsigned char). This is an encrypted access key to unlock the algorithm DLL.

  Return Value:
  true if the DLL is unlocked successfully, false otherwise.

- bool Povolit32(unsigned char EncryptedDllKey[32])
  The purpose of this function is to unlock the security algorithm DLL for specific Fixed Bytes. It uses an encrypted **16** byte access key. This function must be invoked prior to requesting the security key. If this function is not invoked first, the security key requested will always be zero. The access keys can be requested through the Ford NetCom group.
  Note that it is only necessary to call this function once during the application execution. This call shall be located in an area of code (e.g., startup) separate from where GetKey is called and shall not be repeated every time prior to calling GetKey.

  Parameter(s):
  EncryptedDllKey: An array of 32 bytes (unsigned char). This is an encrypted access key to unlock the algorithm DLL.

  Return Value:
  true if the DLL is unlocked successfully, false otherwise.

- int GetKeyAsInt(unsigned char S1, unsigned char S2, unsigned char S3, unsigned char * FixedBytes)
  This function is used to return the securityKey value as a 32 bit integer.

  Parameters:
  S1, S2, S3 are the three random securitySeed bytes returned from the ECU.
  FixedBytes is the array that holds the five fixed challenge bytes.

  Return Value:
  32 bit integer.

- unsigned char GetKey(unsigned char * SecuritySeed, int SecuritySeedLength, unsigned char * FixedBytes, int FixedBytesLength, int AlgorithmType, unsigned char * SecurityKey);

  This function is used to obtain the security key value.

  Parameters:
  SecuritySeed is the 16 byte random securitySeed bytes returned from the ECU.
  SecuritySeedLength is the number of the securitySeed bytes.
  FixedBytes is the array that holds the unencrypted challenge bytes.
  FixedBytesLength is the number of the FixedBytes.
  AlgorithmType is the Ford algorithm type. 1 = Diagnostic Application Security Algorithm, 2 = Diagnostic Programming Security Algorithm.
  SecurityKey is the byte array that will contain the calculated securityKey. The calling application must allocate 16 bytes for Diagnostic Application Security Algorithm key or 18 bytes for Diagnostic Programming Security Algorithm key.

  Return Value:
  1 : Success and the key is returned in SecurityKey parameter.

- unsigned char DecryptBytes(unsigned char *ByteEncrypted , int ByteEncryptedLength, unsigned char *ByteDecrypted)

  This function is used to decrypt a byte array using one of the Ford decryption algorithms.

  Parameters:
  ByteEncrypted is a 16-byte encrypted input value.
  ByteEncryptedLength is the length/size of ByteEncrypted
  ByteDecrypted is the decrypted output value

  Return Value:
  0 : Failure (ByteDecrypted will be all zeros)
  1, 2, 3, 4, 5, 6, 7: This corresponds to the following encryption algorithms (in order): algorithm1, algorithm2, algorithm3, development application, development programming, production application, production programming.

- unsigned char DecryptBytesWithFordAlgorithm(int Alg, unsigned char *ByteEncrypted , int ByteEncryptedLength, unsigned char *ByteDecrypted)

  This function is used to decrypt a byte array using one of the Ford decryption algorithms.

  Parameters:
  Alg is the Ford algorithm to decrypt with (1, 2, 3, ...)
  ByteEncrypted is a 16-byte encrypted input value.
  ByteEncryptedLength is the length/size of ByteEncrypted
  ByteDecrypted is the decrypted output value.

  Return Value:
  0 : Failure (ByteDecrypted will be all zeros)
  1, 2, 3, 4, 5, 6, 7: This corresponds to the following encryption algorithms (in order): algorithm1, algorithm2, algorithm3, development application, development programming, production application, production programming.

- unsigned char EncryptBytesWithFordAlgorithm(int Alg, unsigned char * FixedBytes, int FixedBytesLength, unsigned char * ByteEncrypted)

Parameters:
Alg is the Ford algorithm to encrypt with (1, 2, or 3)
FixedBytes is the input byte array value to encrypt
FixedBytesLength is the length/size of the FixedBytes
ByteEncrypted is the encrypted output value

Return Value:
0 : Failure (ByteEncrypted will be all zeros)
1, 2, 3, 4, 5, 6, 7: This corresponds to the following encryption algorithms (in order): algorithm1, algorithm2, algorithm3, development application, development programming, production application, production programming.

2.2. Examples:

C or C++

```
/// Initialization …
HINSTANCE dllHandle = NULL;

typedef bool (__cdecl* PovolitType)(unsigned char*);
typedef int  (__cdecl* GetKeyType)(unsigned char, unsigned char, unsigned char, unsigned
char *);             // Old Alg (3 fixed bytes)

typedef int  (__cdecl* GetKeyType2)(unsigned char *, int, unsigned char *, int, int,
unsigned char *);    // New Alg (12 fixed bytes)

typedef int  (__cdecl* DecryptBytesType)(unsigned char *, int, unsigned char *);
typedef int  (__cdecl* DecryptBytesType2)(int, unsigned char *, int, unsigned char *);
typedef int  (__cdecl* EncryptBytesType)(int, unsigned char *, unsigned int, unsigned
char *);

// Load the dll and keep the handle to it
dllHandle = LoadLibrary(TEXT("KeyDownWrapper.dll"));
PovolitType PovolitPtr = NULL;
GetKeyType  GetKeyPtr = NULL;
GetKeyType2  GetKeyPtr2 = NULL;

DecryptBytesType DecryptBytesPtr = NULL;
DecryptBytesType2 DecryptBytesPtr2 = NULL;
EncryptBytesType EncryptBytesPtr = NULL;

PovolitPtr = (PovolitType)GetProcAddress(dllHandle, "Povolit");
// If the function address is valid, call the function.
if (PovolitPtr)
{
       unsigned char EncryptedUserData[16] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
       0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
       if (Povolit(EncryptedUserData))
       {
              …
       }
}
```

Obtain the key using the legacy Ford security alogorithm:

```
unsigned char FixedBytesIn[5] = {0x1A, 0x2B, 0x3C, 0x4D, 0x5E};
int key = GetKeyAsInt(0x4E, 0x96, 0x8A, FixedBytesIn);
```

Obtain the key using the new Ford security alogorithm:

```
unsigned char SecuritySeed [16]= {0x11, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88,
0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF};
unsigned char FixedBytes[12] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99,
0xAA, 0xBB, 0xCC};
unsigned char SecurityKeyReturned[18]= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

unsigned char ret = GetKey(SecuritySeed, 16, FixedBytes, 12, 2, SecurityKeyReturned);
```

Encrypt Data:

```
unsigned char FixedBytes[16] = {0xA1, 0xB1, 0xC1, 0xD1, 0xE1, 0x1F, 0x10, 0x11, 0x12,
0x13, 0x14, 0x13};
unsigned char ByteEncrypted[16] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
EncryptBytesPtr = (EncryptBytesType)GetProcAddress(dllHandle,
"EncryptBytesWithFordAlgorithm");
if (EncryptBytesPtr)
{
	unsigned char ret = EncryptBytesPtr(6, FixedBytes, 12, ByteEncrypted);
}
```

Decrypt Data:

```
unsigned char FixedBytes1[16] = {0xA1, 0xB1, 0xC1, 0xD1, 0xE1, 0x1F, 0x10, 0x11, 0x12,
0x13, 0x14, 0x13};
unsigned char ByteEncrypted[16] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char ret = EncryptBytesPtr(6, FixedBytes1, 12, ByteEncrypted);

DecryptBytesPtr = (DecryptBytesType)GetProcAddress(dllHandle, "DecryptBytes");
if (DecryptBytesPtr)
{
	unsigned char ret = DecryptBytesPtr(ByteEncrypted, 16, ByteDecrypted);
}
```