Der AES Algorithmus in LibreOffice Basic

By
Andreas Gronemeyer
getObject.de

Sprache

Germany_De

Der AES Algorithmus in LibO Basic

Bevor Sie den Algorithmus nutzen können müssen Sie die Makrobibliothek **AES** in die Makroverwaltung der Basic IDE importieren. Ich verweise hier auf die **Handbücher von LibreOffice**.

Als bevorzugte Methode können Sie das AES Modul als **Extension bei LibreOffice** downloaden. So bleiben Sie immer auf den neuesten Stand.

Die Verschlüsselungsroutine verläuft nach dem Muster (Pseudocode):

```
erstelleTabellen(LogTable,ExpTable)
erstelleSbox()
erzeugeAESinstanz(meinSchluessel)
verschlüsseln(meinKlarText, meinGeheimText)
```

Die Übergabe der Werte, Schlüssel, der Klartext und der Geheimtext erfolgt über drei Array's. Der zu übergebene Block (Klartext) ist immer 16 Bytes groß. Ist der Block kleiner als 16 Bytes wird dieser intern mit NULL aufgefüllt. Als Ergebnis wird der verschlüsselte Block, auch wieder 16 Bytes, über das dritte Array zurückgegeben.

```
REM die drei Arrays
dim meinSchluessel() as byte              'Leeres Array
dim meinKlarText(15) as byte              'Feste Größe 16 Bytes 0 bis 15
dim meinGeheimText(15) as byte            'Feste Größe 16 Bytes 0 bis 15
```

Vorbereitung zur Ver- oder Entschlüsselung

Bevor eine AES - Instanz erzeugt werden kann müssen die Sbox, Exponential- und Logarithmustabellen bereitgestellt werden. Die Tabellen werden "On the Fly" berechnet. Die Berechnungen übernimmt die Routine

AES.Module1.Main.

Vorab sollte geprüft werden ob AES auch wirklich geladen ist.

```
Sub Main
....
if NOT GlobalScope.Basiclibraries.isLibraryLoaded("AES") then
            GlobalScope.BasicLibraries.LoadLibrary("AES")
end if

AES.Module1.Main

end sub
```

Es liegen die Tabellen nun intern in 4 Array's mit jeweils 256 Bytes vor.

Erzeugung einer AES – Instanz

Um eine AES Instanz zu erzeugen wird der geheime Schlüssel benötigt. In unserem Beispiel übergeben wir den Schlüssel als String um ihn anschließend in das Array `meinSchluessel` mit den zugehörigen Byte werten (Basic: Cbyte) abzulegen.

Der Aufruf zum Erzeugen der AES – Instanz erfolgt durch

```
createAESinstanz( Schlüsselgröße, 1 , meinSchlüssel, Schlüssellänge ).
```

Die Schlüsselgröße gibt die Verschlüsselungstechnik an. Das sind die Modi Bits128 oder Bits192. Die Modi sind im Beispielcode als Konstanten hinterlegt.

```
option explicit
Const Bits128 = 16
Const Bits192 = 24
```

Die 1 als zweiter Parameter wird intern benutzt und ist fest vorgegeben. Der dritte Parameter ist das Array mit dem abgelegten Schlüssel.

Sollte die Schlüssellänge, der vierte Parameter, kleiner als 128 Bits sein, dann wird der Schlüssel intern durch Berechnung aufgefüllt.

Verschlüsseln

Die Verschlüsselung erfolgt durch den Aufruf

```
cipher( meinKlarText, meinGeheimText ).
```

Es ist zu beachten das die zu verschlüsselnde Blockgröße maximal 16 Bytes beträgt. Wenn eine komplette Datei verschlüsselt werden soll muss dies durch eine Schleife erfolgen.

Das Array meinGeheimText enthält nun den verschlüsselten Code. Es ist davon abzuraten diesen als String zu speichern da er nicht druckbare Zeichen enthalten kann.

Entschlüsseln

Das Entschlüsseln ist bei den Vorbereitungen identisch in der Vorgehensweise wie beim Verschlüsseln. Der Pseudocode:

```
erstelleTabellen(LogTable,ExpTable)
erstelleSbox()
erzeugeAESinstanz(meinSchluessel)

REM Hier der einzige Unterschied
entschlüsseln(meinGeheimText, meinKlarText)
```

Die Entschlüsselung wird durch den Subroutine

```
deCipher(meinGeheimText, meinKlarText)
```

durchgeführt. MeiKlarText enthält jetzt den lesbaren Text.

Beispielcode

Diese Datei enthält Basiccode um die Arbeitsweise mit AES zu verdeutlichen. Über eine Dialog Box werden die Eingaben wie der zu verschlüsselnde Text und Ihr Schlüssel verarbeitet.

In der Ausgabezeile erhält man verschlüsselten Text im Hexadezimalformat.

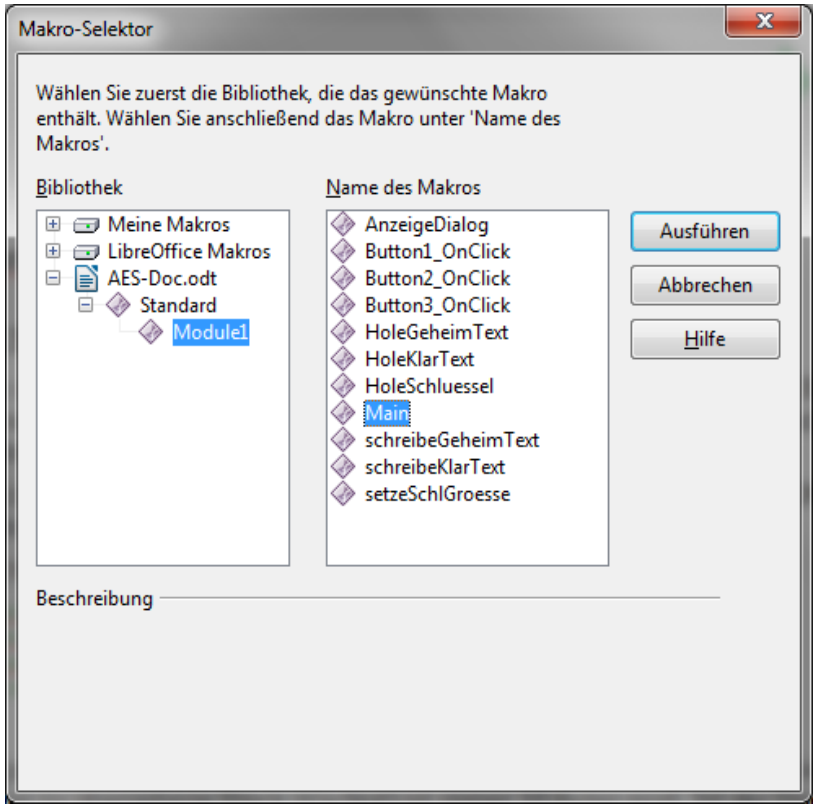
Gestartet wird die Anwendung über das Menü

Extras > Makros > Makro ausführen.

Wenn das Auswahlfenster geöffnet ist, wählen Sie den Eintrag

AES > Standard > Module1 > Main

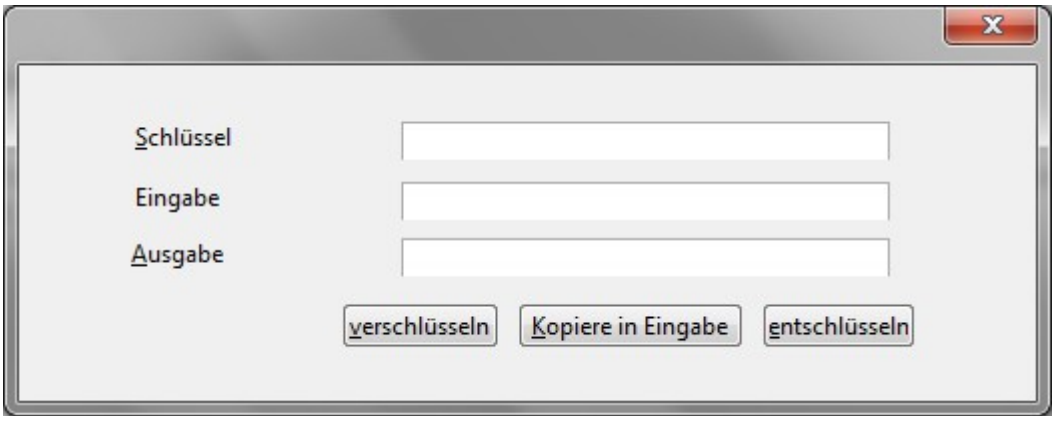
Anschließend startet man das Makro über den Button Ausführen.



Zunächst wird geprüft ob AES geladen ist. Wenn ja werden die oben beschriebenen Tabellen erzeugt.

```
Sub Main
REM Überprüfe ob AES vorhanden ist.
REM Wenn ja dann Bibliothek laden.
if NOT GlobalScope.Basiclibraries.isLibraryLoaded("AES") then
            GlobalScope.BasicLibraries.LoadLibrary("AES")
end if
if NOT ThisComponent.Dialoglibraries.isLibraryLoaded("Standard") then
            ThisComponent.DialogLibraries.LoadLibrary("Standard")
```

```
end if
REM Erzeugen der SBox, Exponential- und Logarithmustabelle
AES.Module1.Main
REM Aufruf der DialogBox
AnzeigeDialog
End Sub
```



Die Erzeugung der AES Instanz und der Aufruf zum Verschlüsseln erfolgt in dem Makro `Button1_OnClick`.

Hier der Code:

```
sub Button1_OnClick
dim meinSchluessel() as byte
dim meinKlarText(15) as byte                    'Feste Größe 16 Bytes 0-15
dim meinGeheimText(15) as byte                  'Feste Größe 16 Bytes 0-15
dim Slaenge as Integer                          'Schlüssellänge
dim keySize as integer                          'Schlüsselgröße
REM Eingabedaten lesen
HoleSchluessel(meinSchluessel)
HoleKlarText(meinKlarText)
REM Schlüsselinstanz vorbereiten
Slaenge = Ubound(meinSchluessel)
REM AES Instanz erzeugen
keySize = setzeSchlGroesse( Slaenge )
createAESinstanz(keySize,1,meinSchluessel,Slaenge)
REM Jetzt verschlüsseln
cipher(meinKlarText, meinGeheimText)
schreibeGeheimText(meinGeheimText)
```

```
end Sub
```

Der Code zum Aufruf der Entschlüsselung befindet sich im Makro Button3_OnClick und ist fast identisch wie die Verschlüsselungsroutine.

Der Code:

```
sub Button3_OnClick
dim meinSchluessel() as byte
dim meinKlarText(15) as byte                    'Feste Größe 16 Bytes 0-15
dim meinGeheimText(15) as byte                  'Feste Größe 16 Bytes 0-15
dim Slaenge as Integer                          'Schlüssellänge
dim keySize as integer                          'Schlüsselgröße
HoleSchluessel(meinSchluessel)
HoleGeheimText(meinGeheimText)
REM Schlüsselinstanz vorbereiten
Slaenge = Ubound(meinSchluessel)
REM AES Instanz erzeugen
keySize = setzeSchlGroesse( Slaenge )
createAESinstanz(keySize,1,meinSchluessel,Slaenge)
REM Jetzt entschlüsseln
deCipher(meinGeheimText, meinKlarText)
schreibeKlarText(meinKlarText)
end Sub
```

English_En

Translated with Google

The AES algorithm in LibO Basic

Before you can use the algorithm, you must import the **AES** macro library into the macro management of the Basic IDE. I refer here to the **LibreOffice manuals**.
The preferred method is to download the AES module as an **extension from LibreOffice**. So you always stay up to date.
The encryption routine runs according to the pattern (pseudocode):

```
erstelleTabellen(LogTable,ExpTable)
erstelleSbox()
erzeugeAESinstanz(meinSchluessel)
verschlüsseln(meinKlarText, meinGeheimText)
```

The transfer of values, keys, plain text and ciphertext takes place via three arrays. The block to be transferred (plain text) is always 16 bytes in size. If the block is smaller than 16 bytes, it is filled internally with NULL. As a result, the encrypted block, again 16 bytes, is returned via the third array.

```
REM the three Arrays
dim meinSchluessel() as byte              'empty Array
dim meinKlarText(15) as byte              'fixed size 16 Bytes 0 to 15
dim meinGeheimText(15) as byte            'fixed size 16 Bytes 0 to15
```

Preparation for encryption or decryption

Before an AES instance can be created, the Sbox, exponential and logarithmic tables must be provided. The tables are calculated "on the fly".
The routine does the calculations

AES.Module1.Main.

You should first check whether AES is actually loaded.

```
Sub Main
....
if NOT GlobalScope.Basiclibraries.isLibraryLoaded("AES") then
            GlobalScope.BasicLibraries.LoadLibrary("AES")
end if

AES.Module1.Main

end sub
```

The tables are now available internally in 4 arrays with 256 bytes each.

Creation of an AES instance

The secret key is required to create an AES instance. In our example we pass the key as a string and then put it in the array `meinSchluessel` with the associated byte values (Basic: Cbyte).
The call to create the AES instance is made by

```
createAESinstanz( Schlüsselgröße, 1 , meinSchlüssel, Schlüssellänge ).
```

The key size indicates the encryption technique. These are the modes Bits128 or Bits192. The modes are stored as constants in the example code.

```
option explicit
Const Bits128 = 16
Const Bits192 = 24
```

The 1 as the second parameter is used internally and is fixed. The third parameter is the array with the stored key.
If the key length, the fourth parameter, is less than 128 bits, then the key is filled up internally by calculation.

Encrypt

The encryption takes place through the call

```
cipher( meinKlarText, meinGeheimText ).
```

Please note that the block size to be encrypted is a maximum of 16 bytes. If a complete file is to be encrypted, this must be done using a loop.
The `meinGeheimText` array now contains the encrypted code. It is not advisable to save this as a string as it can contain non-printable characters.

Decrypt

The decryption is identical to the preparation procedure as for the encryption. The pseudocode:

```
erstelleTabellen(LogTable,ExpTable)
erstelleSbox()
erzeugeAESinstanz(meinSchluessel)
REM Here is the only difference
entschlüsseln(meinGeheimText, meinKlarText)
```

The decryption is done by the subroutine

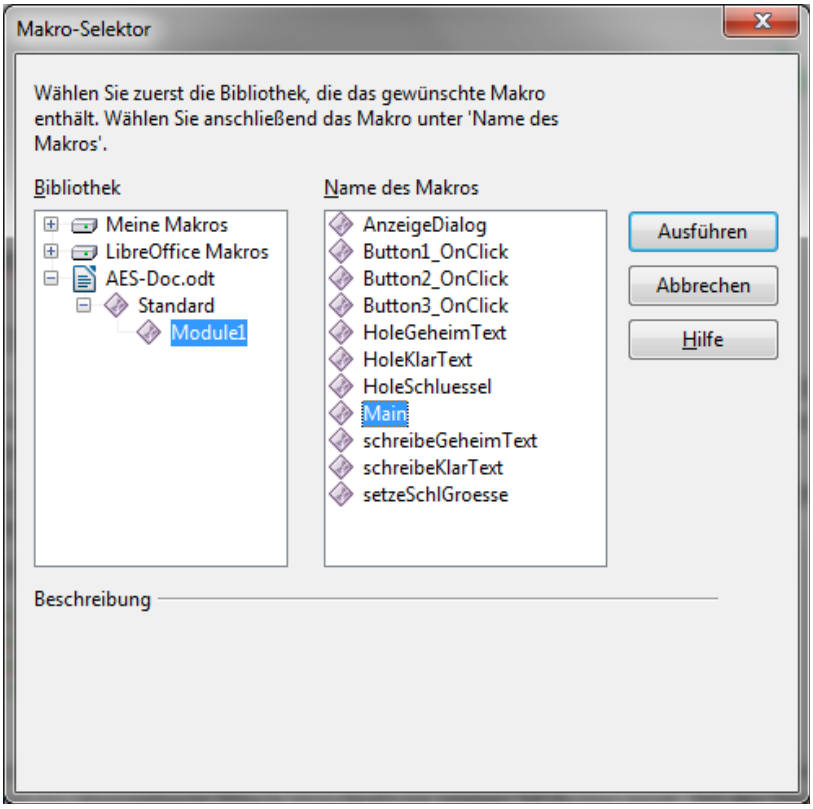
```
deCipher(meinGeheimText, meinKlarText)
```

carried out. `meinKlarText` now contains the readable text.

Sample code

This file contains basic code to clarify how AES works. Entries such as the text to be encrypted and your key are processed via a dialog box.
In the output line you get encrypted text in hexadecimal format.
The application is started via the menu
Tools> Macros> Run Macro.
When the selection window is open, select the entry
AES> Standard> Module1> Main

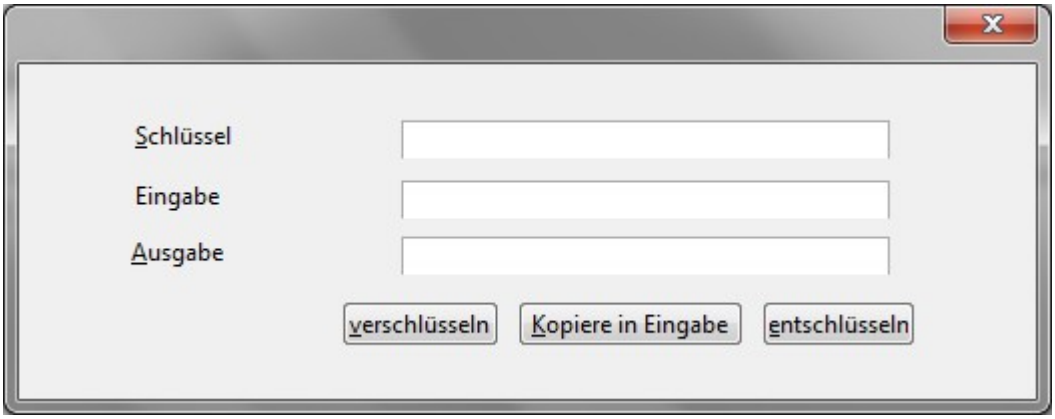
When the selection window is open, select the entry
AES> Standard> Module1> Main
Then you start the macro using the Execute button.



First it is checked whether AES is loaded. If so, the tables described above are generated.

```
Sub Main
REM Überprüfe ob AES vorhanden ist.
REM Wenn ja dann Bibliothek laden.
if NOT GlobalScope.Basiclibraries.isLibraryLoaded("AES") then
            GlobalScope.BasicLibraries.LoadLibrary("AES")
end if
if NOT ThisComponent.Dialoglibraries.isLibraryLoaded("Standard") then
            ThisComponent.DialogLibraries.LoadLibrary("Standard")
end if
REM Erzeugen der SBox, Exponential- und Logarithmustabelle
AES.Module1.Main
```

```
REM Aufruf der DialogBox
AnzeigeDialog
End Sub
```



The AES instance is generated and the encryption is called in the macro `Button1_OnClick`.

Here is the code:

```
sub Button1_OnClick
dim meinSchluessel() as byte
dim meinKlarText(15) as byte                    'Feste Größe 16 Bytes 0-15
dim meinGeheimText(15) as byte                  'Feste Größe 16 Bytes 0-15
dim Slaenge as Integer                          'Schlüssellänge
dim keySize as integer                          'Schlüsselgröße
REM Eingabedaten lesen
HoleSchluessel(meinSchluessel)
HoleKlarText(meinKlarText)
REM Schlüsselinstanz vorbereiten
Slaenge = Ubound(meinSchluessel)
REM AES Instanz erzeugen
keySize = setzeSchlGroesse( Slaenge )
createAESinstanz(keySize,1,meinSchluessel,Slaenge)
REM Jetzt verschlüsseln
cipher(meinKlarText, meinGeheimText)
schreibeGeheimText(meinGeheimText)
end Sub
```