# A Fast and Easy to Use **AES** Library

**robertguan**, 28 Dec 2011 CPOL

4.82 (32 votes)

Rate:

A fast and easy to use **AES** library.

**Download source code v1.0 - 1.0 MB**

**Download source code v2.0 - 321 KB**

## Introduction

Ef**Aes**Lib is a highly optimized Advanced Encryption Standard (**AES**) library for the Windows platform 32-bit architecture. The Extreme Fast **AES** Library is implemented based on the official document: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

The library is actually my personal work. I have decided to put it in the public domain and make it free. The size is a little on the higher side because of some optimization to use space in exchange of time.

I have provided the compiled DLL in VS2008, and the project files; or you can use the source in any other platform, it is just plain 'C'.

# Using the code

**AES** is a 128-bit block encrypt/decrypt algorithm. That means you need to carefully handle the last block which is not 16 bytes aligned. Otherwise, you might be unable to decrypt correctly.

There are many block modes defined in the cipher realm. Different block modes have different characteristics. For example, the CRT mode only needs encryption logic, so it is suitable for low cost hardware implementations. The PCBC mode provides better error propagation. As for CFB, OFB modes, there is an extra parameter: 'feedback size'. You can treat it as the result size of each **AES** block process. That means, CFB 8-bits mode should be about 16 times slower than CFB 128-bits mode. And also, you can do stream ciphers by using the CFB 8-bits mode.

You can reference the *Ef**Aes**Lib.pdf* in the package for details about how the different block modes work.

| | Encode/Decode with same process | Need Initial Vector | Chain process |
|---|---|---|---|
| ECB | X | X | X |
| CBC | X | O | O |
| PCBC | X | O | O |
| CFB | O | O | O |
| OFB | O | O | O |
| CRT | O | O | O |

**AES** always needs a 128-bit key to encrypt/decrypt. But it is also combined with an initial vector to work with, except in ECB mode. Each bit of the initial vector you use will double the possibilities of encrypted text from a given plain text, which means more safety.

Ef**Aes**Lib supports ECB, CBC, PCBC, OFB, CFB, CRT block modes, and support OFB,CFB mode with [1..16] bytes feedback size. It also supports in-place encryption/decryption in each mode (source and destination buffer are the same).

The following sample uses Counter mode to encode a file:

Hide Shrink ▲ Copy Code

```
#include "EfAes.h"
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
```

```
#include <stdlib.h>
int main(int argc , char * argv[])
{
    unsigned char key[16]={
       0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,
       0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88
    };
    unsigned char vector[16]={
       0x1f,0x32,0x43,0x51,0x56,0x98,0xaf,0xed,
       0xab,0xc8,0x21,0x45,0x63,0x72,0xac,0xfc
    };
    unsigned char buff[4096];
    int rd_fd,wr_fd, rdsz;
    AesCtx context;
    AesSetKey( &context , AES_KEY_128BIT ,BLOCKMODE_CRT, key , vector );
    rd_fd = open("test.dat", O_RDONLY);
    wr_fd = open("test.encoded",O_WRONLY | O_CREAT);
    setmode(rd_fd,O_BINARY);
    setmode(wr_fd,O_BINARY);
    while( (rdsz = read(rd_fd, buff ,4096)) > 0 )
    {
      // before last block , the block size
      // should always be the multiply of 16
      // the last block should be handled
      // if the size is not a multiply of 16
      AesEncryptCRT(&context , buff, buff, rdsz );
      rdsz = AesRoundSize( rdsz, 16);
      write( wr_fd , buff , rdsz );
    }
    close(rd_fd);
    close(wr_fd);
}
```

The use of the `AesCtx` structure is mainly designed for thread issues. Each encryption session should have its own `AesCtx`. The Ef**Aes**Lib APIs will always pad 0 to input data whose size is not a multiple of 16, or a multiple of the feedback size in the CFB, OFB modes.

# Optimization

There are pre-defined functions in the **AES** algorithm. The first step, also proposed in the Wiki, is to combine `SubBytes`, `ShiftRows` with `MixColumns`. The follow is my sample implementation:

Hide Copy Code

```
void SubAndShiftAndMixRound(uint8 * pState ,uint32 * pRoundKey , uint32 * pOutput)
{
    uint32 a1,a2,a3,a4;

    a1=pState[0];
```

```
    a2=pState[5];
    a3=pState[10];
    a4=pState[15];

    *pOutput++ =
        ((SboxXTime2[a1] ^ SboxXTime3[a2] ^        FSB[a3] ^
                                                   FSB[a4])        |
              ((FSB[a1] ^ SboxXTime2[a2] ^ SboxXTime3[a3] ^
                                           FSB[a4]) << 8)  |
              ((FSB[a1] ^        FSB[a2] ^ SboxXTime2[a3] ^
                                 SboxXTime3[a4]) << 16 )|
        ((SboxXTime3[a1] ^       FSB[a2] ^        FSB[a3] ^
                                 SboxXTime2[a4]) << 24))^ *pRoundKey++;

    ............
}
```

In the second step, notice the horizontal direction of a1, a2, a3, a4. We can reduce this by using a pre-build lookup table for each column.

Hide Copy Code

```
for(i=0;i<256;i++)
{
    TestTable1[i]=SboxXTime2[i] | FSB_8[i]        | FSB_16[i]        | SboxXTime3_24[i];
    TestTable2[i]=SboxXTime3[i] | SboxXTime2_8[i] | FSB_16[i]        | FSB_24[i];
    TestTable3[i]=FSB[i]        | SboxXTime3_8[i] | SboxXTime2_16[i] | FSB_24[i];
    TestTable4[i]=FSB[i]        | FSB_8[i]        | SboxXTime3_16[i] | SboxXTime2_24[i];
}
```

The code in step one will be optimized to:

Hide Copy Code

```
void SubAndShiftAndMixRound(uint8 * pState ,uint32 * pRoundKey , uint32 * pOutput)
{
    uint32 a1,a2,a3,a4;

     a1=pState[0];
     a2=pState[5];
     a3=pState[10];
     a4=pState[15];

     *pOutput++ = TestTable1[a1] ^ TestTable2[a2] ^ TestTable3[a3] ^
                                  TestTable4[a4] ^ *pRoundKey++;
     ............
}
```

In the third step, notice `a1=pState[0],a2=pState[5],a3=pState[10],a4=pState[15]`; it is slow in the 32-bit architecture. We can change it to a 32-bit access and XOR the sequence.

# Performance

The best performance Ef**Aes**Lib has is 10M bytes in 78 milliseconds with my Pentium IV 3.0Ghz computer.

# Reference

The official document: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

# The Wiki

- http://en.wikipedia.org/wiki/**AES**
- http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

# History

v2.0: Extended the library to 128/192/**256** bits key length, and also added a 64 bit DLL in addition.

# License



# Share



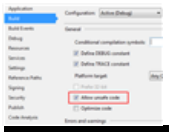
# About the Author

**robertguan**

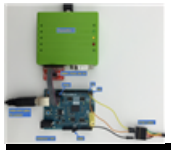
Engineer I will tell you after my death.
Taiwan

No Biography provided

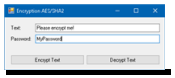
## You may also be interested in...



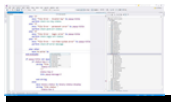
C# AES 256 bits Encryption Library with Salt



Get Started on Arduino 101/Genuino 101 with Intel® System Studio for Microcontrollers using the Flyswatter 2 JTAG Debugger



Encrypt Strings with Passwords - AES 256 & SHA256



10 Ways to Boost COBOL Application Development


Visual COBOL New Release: Small point. Big deal

SAPrefs - Netscape-like Preferences Dialog

## Comments and Discussions

Add a Comment or Question

Search Comments Go

Spacing Relaxed Layout Open All Per page 25 Update

First Prev Next

## Compilation difficulties

Cameron Caturria 31-Jan-16 5:46

Hello.
I wanted to document my efforts so far in compiling this library. Suffice it to say that for now it's definitely not going to compile out of the box -- and using the prebuilt DLL is not an option for me as my project is cross platform and I prefer to static link (or better yet imbed libraries in to my project directly, when possible).
First of all: the stdafx.h include; probably should be removed because different compilers provide different vehicles for precompiled headers.

Byte is not defined so I assumed char and typedef'd it accordingly in AesInterns.h.

#include <string> so lib can find memcpy and friends.

This is where things became difficult. AesEncode.cpp seems to have been butchered in transit. On line 2487 begins a duplicate of previous code beginning somewhere in the middle of the initializer list for RevRawTable3. From there, RevRawTable4 is duplicated and some of the next functions which were defined after it.
If you still have a version of this project which is not damaged, could you please consider reuploading?
Thanks.

Reply · Email · View Thread · Permalink · Bookmark

## There are a number of bugs in version 2.0

Ben321 7-Jan-16 11:00

I wouldn't have worried about it, except I found out when using the DLL file in VB6 that it was using CDECL calling convention for all its exported functions (which Visual Basic 6 can't use). This meant I had to compile a copy of the DLL file (for which I used Visual C++ 2010 Express) myself, and do so in STDCALL mode. Unfortunately as soon as I compiled it, it started throwing both warnings and errors. I finally fixed all the code bugs, or at least all the ones that triggered warnings and errors. After compiling the DLL file, I then tested it in a VB6 program. It seems to work properly, giving the expected outputs for various inputs. It is able to properly encrypt and decrypt data given to it. But I'm still not completely sure that it's 100% bug free. I noticed that after a time using it, I started getting random "out of memory" errors in the VB6 IDE when I tried to add a new module or form to the project. I'm not sure if that's related to this DLL, or something else, but it's possible that the DLL still has some bugs that don't trigger warnings or errors when compiling, that end up corrupting the memory space of any program that uses the DLL, or creates memory leaks, or other similar issues. I'll later on upload this fixed (mostly) source code (which I'll call version 2.1), complete with the VC++2010 project file, so nobody else will need to take time to configure their VC++ project for compiling the source code like I had to. I'll upload it to Mediafire, and then post a link to download it here in the comments section.

Maybe somebody else in the community will be able to finish fixing the source code, and be able to post a completely fixed version 2.2 of the code.

Reply · Email · View Thread · Permalink · Bookmark

## Re: There are a number of bugs in version 2.0

Ben321 7-Jan-16 11:22

Ok, just uploaded my version 2.1 to Mediafire. Here's the link, just as I promised.
https://www.mediafire.com/?s8yujbdypnn1y8e

If you look inside the zip file, insite the EfAes folder and look in the Release folder, you will see that it already has the compiled DLL file there. That way you can use it, even if you don't want to compile it. As I said though it still may have some bugs in it. So I highly encourage others in the development community to thoroughly check over the code and make sure that everything is correct, and fix any additional bugs that may be present.

Reply · Email · View Thread · Permalink · Bookmark

### What's the AesCtx * pContext parameter?

Ben321 6-Jan-16 22:36

In every one of these AES functions, there's this AesCtx * pContext parameter. I get that it's a pointer to something called pContext, of type AesCtx, but how do I get this value to start with? Normally when there's something like a file handle (hFile) you first call OpenFile to get the handle, then you do stuff with that handle, and then you end it by calling CloseHandle. But in the exported functions from your AES DLL, I don't see any Open or Close functions. Please tell me how I am supposed to get this pContext to start with, since there are no obvious handle opening or closing functions in this DLL file.

Reply · Email · View Thread · Permalink · Bookmark

### Re: What's the AesCtx * pContext parameter?

Ben321 7-Jan-16 10:44

Ok, after looking over all the source code, I finally figured out what the AesCtx data type is. It's just a 512 byte array that is intented for internal use by the DLL file. Using the DLL file in programming languages other than C or C++, you just need to make an empty 512 byte array and give the pointer to this array to the pContext parameter in any functions that require it.

Reply · Email · View Thread · Permalink · Bookmark

### spot on.

KenSands 30-Oct-14 3:50

If only Microsoft's .net crypto stuff worked as well as this.

Reply · View Thread · Permalink · Bookmark

### not "just plain 'C'."

Member 10443095 4-Dec-13 12:18

"you can use the source in any other platform, it is just plain 'C'."

this code is very visual studio specific for example "typedef unsigned __int64 uint64;" __int64 as far as i know is a Visual studio type only(never found another compiler on win/linux that has it). for gcc it would be "typedef unsigned long long uint64;" older versions of VS did not support long long (i remember having this issue before) but it is there in VS2010+ not sure when MS added long long but it is more widely supported.

also file names do not match includes in a non windows environments case matters #include "AESInterns.h" will not find AesInterns.h and it is not just a matter of changing the file name because #includes are not consistent.

_WIN64 is not defined on other platforms see this article for other approaches http://stackoverflow.com/questions/735647/ifdef-for-32-bit-platform

lastly __declspec with gcc you get the following error "ISO C++ forbids declaration of '__declspec' with no type" here is an article on that http://stackoverflow.com/questions/7895869/cross-platform-alignx-macro at this point i gave up on the port and am looking for another option i may come back if i dont find one.

Reply · Email · View Thread · Permalink · Bookmark

## Good

microbio75 **7-Oct-13 16:12**

Good job, It's very useful to me,

---

Regards

Juan

Reply · Email · View Thread · Permalink · Bookmark

## buggy v2.0

PepekNamornik **1-Dec-12 21:03**

I can confirm that v2.0 sources are messy and cant be compiled in the original form. It should be put in order by the author. I tried DecryptPCBC (in_place) and it behaves differently in WinXP and in Win7... only first block (128 bytes) is properly decoded in Win7. Source code need to be rewritten!

Reply · View Thread · Permalink · Bookmark 1.00/5 (1 vote)

## Re: buggy v2.0

Ben321 6-Jan-16 22:39

Use the already existing DLL file that is included in the ZIP archive. There's no need to recompile it. The source code is just there in case you want to tinker with it. There's no guarenty it will work as is. And it doesn't need to work either. You just need the DLL file that's in the lib folder, as that is the already compiled DLL file.

Reply · Email · View Thread · Permalink · Bookmark

## My vote of 5

sukumarchandran **25-Oct-12 18:20**

unfortunately i miss understood at first but its really awesome code

Reply · Email · View Thread · Permalink · Bookmark

## AESEncode.cpp seems to be corrupt

RobsterNZ 12-Sep-12 8:37

Hi robertguan,

Thanks for this article - really good work, and especially nice that you've taken the time to provide some documentation!

A couple of comments:
1) I can use your DLL, but I can't compile the project because AESEncode.cpp isn't all there. If you look around line 2487, there's an incomplete line in the declaration of a data table (presumably RevRawTable3). Could you please look into that?

2) In section 3.1 of the PDF, which describes AesSetKey, the comment for pKey is "Pointer to 128 bit key". Should this be changed to "pointer to key where length is determined by AesKeyLength" or something similar?

Cheers,
Rob.

Reply · Email · View Thread · Permalink · Bookmark 5.00/5 (4 votes)

## Re: AESEncode.cpp seems to be corrupt

microbio75 7-Oct-13 16:07

Hi Rob.

These tables are repeated in the code, so you can delete them and you will be able to compile the code.

---

Regards.

Juan

Reply · Email · View Thread · Permalink · Bookmark

## pretty good

chenzhe62 21-Apr-12 12:37

good

Reply · Email · View Thread · Permalink · Bookmark

## Compilation errors

tinarajraj 7-Feb-12 16:25

I have taken the v2.0 and am unable to compile. There is #if 0 defined in AESencode.c in line 2544 which has no corresponding #endif.

Reply · Email · View Thread · Permalink · Bookmark

## My vote of 5

Monjurul Habib 29-Dec-11 1:10

nice work

Reply · Email · View Thread · Permalink · Bookmark

## aes 256 cbc

zby84722 14-Dec-11 0:46

i need to use aes-256-cbc in my C project.Could this library use? and how to use. In afAesLib.pdf i just know how to use aes-128-cbc.

Reply · Email · View Thread · Permalink · Bookmark

## Extension to use 256 bit keys

MemberPhillip 19-Feb-10 11:24

Thanks Robert.

> maybe you can try to change the Nk=8 defined in AesEncode.cpp
> or you can modify the AesSetKey to make it more flexible.

Unfortunately it takes a bit more tinkering to extend the algorithm to
use 256 bit keys:

- set Nk=8, i.e. key size=8
- set Nr=14, i.e. number of rounds, Nr=Nk+6

The follow on effects of the above parameter settings are:

- update array sizes in the AESContext structure, i.e.

```
typedef struct _AESContext_
{
uint32 AesEncKey[60];
uint32 AesDecKey[60];
```

```
uint32 InitialVector[4];
uint32 iFeedBackSz;
} AESContext;

- update array size in AesCtx structure, i.e.

typedef struct _AesCtx_
{
unsigned char space[500];
} AesCtx;

- update the AES_ENCRYPT and AES_DECRYPT macros, e.g. for the AES_ENCRYPT
macro insert:

a1=TestTable1[ (byte)state ];\
a4=TestTable2[ (( regs *) &state)->ah ]; \
state = state >> 16;\
a3=TestTable3[ (byte)state ];\
a2=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[1];\
a2^=TestTable1[ (byte)state ];\
a1^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a4^=TestTable3[(byte)state ];\
a3^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[2];\
a3^=TestTable1[ (byte)state ];\
a2^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a1^=TestTable3[ (byte)state ];\
a4^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[3];\
\
a4^=TestTable1[ (byte)state ];\
a3^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a2^=TestTable3[ (byte)state ];\
a1^=TestTable4[ (( regs *) &state)->ah ];\
\
state=a1^AesEncKey[40];\
temp[1]=a2^AesEncKey[41];\
```

```
temp[2]=a3^AesEncKey[42];\
temp[3]=a4^AesEncKey[43];\
\
a1=TestTable1[ (byte)state ];\
a4=TestTable2[ (( regs *) &state)->ah ]; \
state = state >> 16;\
a3=TestTable3[ (byte)state ];\
a2=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[1];\
a2^=TestTable1[ (byte)state ];\
a1^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a4^=TestTable3[(byte)state ];\
a3^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[2];\
a3^=TestTable1[ (byte)state ];\
a2^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a1^=TestTable3[ (byte)state ];\
a4^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[3];\
\
a4^=TestTable1[ (byte)state ];\
a3^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a2^=TestTable3[ (byte)state ];\
a1^=TestTable4[ (( regs *) &state)->ah ];\
\
state=a1^AesEncKey[44];\
temp[1]=a2^AesEncKey[45];\
temp[2]=a3^AesEncKey[46];\
temp[3]=a4^AesEncKey[47];\
\
a1=TestTable1[ (byte)state ];\
a4=TestTable2[ (( regs *) &state)->ah ]; \
state = state >> 16;\
a3=TestTable3[ (byte)state ];\
a2=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[1];\
a2^=TestTable1[ (byte)state ];\
```

```
a1^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a4^=TestTable3[(byte)state ];\
a3^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[2];\
a3^=TestTable1[ (byte)state ];\
a2^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a1^=TestTable3[ (byte)state ];\
a4^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[3];\
\
a4^=TestTable1[ (byte)state ];\
a3^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a2^=TestTable3[ (byte)state ];\
a1^=TestTable4[ (( regs *) &state)->ah ];\
\
state=a1^AesEncKey[48];\
temp[1]=a2^AesEncKey[49];\
temp[2]=a3^AesEncKey[50];\
temp[3]=a4^AesEncKey[51];\
\
a1=TestTable1[ (byte)state ];\
a4=TestTable2[ (( regs *) &state)->ah ]; \
state = state >> 16;\
a3=TestTable3[ (byte)state ];\
a2=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[1];\
a2^=TestTable1[ (byte)state ];\
a1^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a4^=TestTable3[(byte)state ];\
a3^=TestTable4[ (( regs *) &state)->ah ];\
\
state = temp[2];\
a3^=TestTable1[ (byte)state ];\
a2^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a1^=TestTable3[ (byte)state ];\
a4^=TestTable4[ (( regs *) &state)->ah ];\
```

```
\
state = temp[3];\
\
a4^=TestTable1[ (byte)state ];\
a3^=TestTable2[ (( regs *) &state)->ah ];\
state = state >> 16;\
a2^=TestTable3[ (byte)state ];\
a1^=TestTable4[ (( regs *) &state)->ah ];\
\
state=a1^AesEncKey[52];\
temp[1]=a2^AesEncKey[53];\
temp[2]=a3^AesEncKey[54];\
temp[3]=a4^AesEncKey[55];\
\
a1= FSB [(byte) state];\
a4= FSB_8[(( regs *) &state)->ah];\
state = state >> 16;\
a3= FSB_16[(byte) state ];\
a2= FSB_24[(( regs *) &state)->ah];\
\
state = temp[1];\
a2|= FSB [(byte) state];\
a1|= FSB_8[(( regs *) &state)->ah];\
state = state >> 16;\
a4|= FSB_16[(byte) state ];\
a3|= FSB_24[(( regs *) &state)->ah];\
\
state = temp[2];\
a3|= FSB [(byte) state];\
a2|= FSB_8[(( regs *) &state)->ah];\
state = state >> 16;\
a1|= FSB_16[(byte) state ];\
a4|= FSB_24[(( regs *) &state)->ah];\
\
state = temp[3];\
a4 |= FSB [(byte) state]; \
a3 |= FSB_8[(( regs *) &state)->ah]; \
state = state >> 16;\
a2 |= FSB_16[(byte) state ]; \
a1 |= FSB_24[(( regs *) &state)->ah]; \
\
\
pOutput[0]= a1 ^ AesEncKey[56];\
pOutput[1]= a2 ^ AesEncKey[57];\
```

```
pOutput[2]= a3 ^ AesEncKey[58];\
pOutput[3]= a4 ^ AesEncKey[59];
```

Hope this helps others wanting to extend the source code to use a 256 key.

Reply · Email · View Thread · Permalink · Bookmark 1.00/5 (1 vote)

## Re: Extension to use 256 bit keys

MemberPhillip 19-Feb-10 11:33

One other change required is to increase the array sizes in the various
chaining block functions.

That is, replace 44 with 60, e.g. in AesEncryptECB()

```
void AesEncryptECB( AesCtx * pContext , void * pOutput, void * pInput ,int iSize )
{
ALIGN uint32 a1,a2,a3,a4,state;
ALIGN uint32 local_temp[4];
#if 0
ALIGN uint32 local_key[44];
#else
ALIGN uint32 local_key[60];
#endif

iSize = padding( pInput , iSize );

#if 0
memcpy( local_key , ((AESContext*)pContext)->AesEncKey , 44 * 4); // the key point is to declare array type.
#else
memcpy( local_key , ((AESContext*)pContext)->AesEncKey , 60 * 4); // the key point is to declare array type.
#endif

while(iSize>0)
{

AES_ENCRYPT( ((uint32*)pInput), ((uint32 *)pOutput),local_key,local_temp);
//AES_ENCRYPT_TEST( ((uint32*)pInput), ((uint32 *)pOutput),local_key);
//aes_encrypt((uint32*)pInput,(uint32*)pOutput,g_TestKey);

pInput = (uint32 *) pInput + 4;
pOutput = (uint32 *) pOutput + 4;
iSize -= 16;
```

```
}
}
```

A more tidy approach would be to define the array size in a parameter, where the 44 and 60 is equal to Nb*(Nr+1).

Reply · Email · View Thread · Permalink · Bookmark 1.00/5 (1 vote)

## Good Job !

**m.moestl** **12-Feb-10 21:00**

Thanks for publish this!

I want to use it, but I currently use 256-bit keys...

can you show the code for them too ?

thanks, matth

Reply · View Thread · Permalink · Bookmark

### Re: Good Job !

robertguan 12-Feb-10 23:59

maybe you can try to change the Nk=8 defined in AesEncode.cpp
or you can modify the AesSetKey to make it more flexible.

Reply · Email · View Thread · Permalink · Bookmark

## Completeness

**Panic2k3** **11-Feb-10 5:35**

It would be interesting to have a benchmark comparison between other Open Source/PD implementation of the AES algorithm.

/// HACK: Don't PANIC ...
http://en.wikibooks.org/wiki/User:Panic2k4

Reply · View Thread · Permalink · Bookmark

### Re: Completeness

robertguan 11-Feb-10 9:30

I have ever compare it with quickcrypt
so I believe it is in commercial degree,
even in some block mode, my optimize is far better than quickcrypt.
but I would not spend more time to do a list.

Reply · Email · View Thread · Permalink · Bookmark

## my vote of 5

Rozis 10-Feb-10 7:56

Nice, no phony pony class 'state of the art' stuff - just plain C - i like it!

But to be honest: You could give a brief description on the different modes. It will make your article more readable - i think.

I'll definitely try to put your code in my toolbox..

ThanX

Reply · View Thread · Permalink · Bookmark

## Re: my vote of 5

robertguan 10-Feb-10 14:16

people who intend to understand the block process can reference the efAesLib.pdf included in the zip.

Reply · Email · View Thread · Permalink · Bookmark 1.00/5 (1 vote)

Last Visit: 6-Mar-17 10:56 Last Update: 6-Mar-17 11:00 Refresh 1 2 Next »

General News Suggestion Question Bug Answer Joke Praise Rant Admin