

AES

1.1

Направљено помоћу Doxygen 1.8.7

Fri Jun 20 2014 16:18:11



# Садржај

1	Индекс датотека	1
1.1	Списак датотека . . . . .	1
2	Документација датотеке	3
2.1	AES/aes.c Референца датотеке . . . . .	3
2.1.1	Документација функције . . . . .	4
2.1.1.1	AddRoundKey . . . . .	4
2.1.1.2	CRC32 . . . . .	4
2.1.1.3	CRCCheck . . . . .	5
2.1.1.4	CypherAES . . . . .	5
2.1.1.5	CypherFileName . . . . .	6
2.1.1.6	DecryptState . . . . .	6
2.1.1.7	DecypherAES . . . . .	7
2.1.1.8	DeCypherFileName . . . . .	7
2.1.1.9	EncryptState . . . . .	8
2.1.1.10	InvMixColumns . . . . .	8
2.1.1.11	InvShiftRows . . . . .	8
2.1.1.12	InvSubBytes . . . . .	9
2.1.1.13	KeyExpansion . . . . .	9
2.1.1.14	main . . . . .	9
2.1.1.15	MixColumns . . . . .	9
2.1.1.16	ReadData . . . . .	10
2.1.1.17	RotWord . . . . .	10
2.1.1.18	ShiftRows . . . . .	10
2.1.1.19	SizeOfFile . . . . .	11
2.1.1.20	StateToUInt32 . . . . .	11
2.1.1.21	SubBytes . . . . .	11
2.1.1.22	SubWord . . . . .	12
2.1.1.23	UCharToUInt32 . . . . .	12
2.1.1.24	UInt32ToState . . . . .	12
2.1.1.25	WriteData . . . . .	13

2.2	AES/aes.h Референца датотеке . . . . .	13
2.2.1	Опширније . . . . .	15
2.2.2	Документација дефиниције . . . . .	15
2.2.2.1	_aes_h_ . . . . .	15
2.2.2.2	_CRT_SECURE_NO_DEPRECATED . . . . .	15
2.2.2.3	CRYPT_TEXT . . . . .	15
2.2.2.4	DIGIT_NUM . . . . .	15
2.2.2.5	MAX_FILE_PATH . . . . .	15
2.2.2.6	NUM_ROW . . . . .	15
2.2.2.7	STATE_SIZE . . . . .	15
2.2.3	Документација дефиниције типа . . . . .	16
2.2.3.1	UChar . . . . .	16
2.2.3.2	UInt32 . . . . .	16
2.2.4	Документација функције . . . . .	16
2.2.4.1	AddRoundKey . . . . .	16
2.2.4.2	CRC32 . . . . .	16
2.2.4.3	CRCCheck . . . . .	16
2.2.4.4	CypherAES . . . . .	17
2.2.4.5	CypherFileName . . . . .	18
2.2.4.6	DecryptState . . . . .	18
2.2.4.7	DecypherAES . . . . .	18
2.2.4.8	DeCypherFileName . . . . .	19
2.2.4.9	EncryptState . . . . .	19
2.2.4.10	InvMixColumns . . . . .	20
2.2.4.11	InvShiftRows . . . . .	20
2.2.4.12	InvSubBytes . . . . .	20
2.2.4.13	KeyExpansion . . . . .	21
2.2.4.14	MixColumns . . . . .	21
2.2.4.15	ReadData . . . . .	21
2.2.4.16	RotWord . . . . .	22
2.2.4.17	ShiftRows . . . . .	22
2.2.4.18	SizeOfFile . . . . .	22
2.2.4.19	StateToUInt32 . . . . .	23
2.2.4.20	SubBytes . . . . .	23
2.2.4.21	SubWord . . . . .	23
2.2.4.22	UCharToUInt32 . . . . .	24
2.2.4.23	UInt32ToState . . . . .	25
2.2.4.24	WriteData . . . . .	25
2.2.5	Документација променљиве . . . . .	25
2.2.5.1	CRC32_TABLE . . . . .	26

---

2.2.5.2	G_FIELD_MUL . . . . .	26
2.2.5.3	INVS_BOX . . . . .	26
2.2.5.4	ROUND_CONSTANT . . . . .	26
2.2.5.5	S_BOX . . . . .	26



# Глава 1

## Индекс датотека

### 1.1 Списак датотека

Овде је списак свих датотека са кратким описима:

AES/ <a href="#">aes.c</a> . . . . .	3
AES/ <a href="#">aes.h</a>	
Фајл садржи функције за шифровање и дешифровање датотека АЕС-ом . . . . .	13





## Глава 2

# Документација датотеке

### 2.1 AES/aes.c Референца датотеке

```
#include "aes.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <limits.h>
```

#### Функције

- `UInt32 UCharToUInt32 (const UChar byte[])`  
Претвара низ 4 бајта у `UInt32`.
- `void AddRoundKey (UChar state[][4], const UInt32 key[])`  
Врши операцију еклузивне дисјункције блока бајтова величине `STATE_SIZE` и кључа.
- `void SubBytes (UChar state[][4])`  
Мења вредност сваког појединачног бајта из `state` одговарајућом вредношћу из `S_BOX`.
- `void InvSubBytes (UChar state[][4])`  
Мења вредност сваког појединачног бајта из `state` одговарајућом вредношћу из `INVS_BOX`.
- `void ShiftRows (UChar state[][4])`  
Ротира бајтове у врстама, у *i*-тој врсти за *i* места улево.
- `void InvShiftRows (UChar state[][4])`  
Ротира бајтове у врстама, у *i*-тој врсти за *i* места удесно.
- `void MixColumns (UChar state[][4])`  
Миксује колоне матрице бајтова `state`.
- `void InvMixColumns (UChar state[][4])`  
Инверзно миксује колоне матрице бајтова `state`.
- `void RotWord (UInt32 *wordPt)`  
1-бајтна кружна ротација 4-бајтне(`UInt32`) речи улево.
- `UInt32 SubWord (const UInt32 word)`  
Замена сваког бајта 4-бајтне речи `word` одговарајућим бајтом из `S_BOX`.
- `int KeyExpansion (const UChar key[], UInt32 **word, const int keySize)`  
Развија кључ `key` у низ 4-бајтних кључева `w`.
- `void EncryptState (const UChar in[], UChar out[], const UInt32 key[], const int keySize)`  
Шифрује АЕС-ом низ бајтова величине `STATE_SIZE` задат са `in` помоћу низа кључева `key` (добijenih `KeyExpansion`) и уписује у `out`.

- void **DecryptState** (const **UChar** in[], **UChar** out[], const **UInt32** key[], const int keySize)  
Дешифрује АЕС-ом низ бајтова величине **STATE\_SIZE** задат са in помоћу низа кључева key (добитених **KeyExpansion**) и уписује у out.
- **UInt32** **SizeOfFile** (const **UChar** \*name)  
Враћа величину датотеке имена name која се налази у истом фолдеру у којем је програм.
- int **ReadData** (FILE \*f, **UChar** state[], **UInt32** \*sizeToEnd)  
Чита из датотеке f блок бајтова **STATE\_SIZE** и смешта у state.
- int **WriteData** (FILE \*f, const **UChar** state[])  
Пише у датотеку f блок бајтова **STATE\_SIZE** из state.
- void **UInt32ToState** (const **UInt32** num, **UChar** state[])  
Преводи 4-бајтни број типа **UInt32** у низ од 16 бајтова.
- **UInt32** **StateToUInt32** (const **UChar** state[])  
Преводи низ од 16 бајтова у број од 4 бајта (**UInt32**).
- **UChar** \* **CypherFileName** (const **UChar** \*inFileName)  
Прави име шифроване датотеке од имена датотеке који треба да шифрује
- int **DeCypherFileName** (**UChar** \*\*inFileName)  
Враћа име датотеке које ће имати након што се дешифрује.
- **UInt32** **CRC32** (**UInt32** crc, const **UChar** state[])  
Враћа вредност CRC-а датог блока state од **STATE\_SIZE** бајтова и crc.
- int **CRCCheck** (FILE \*in, const **UInt32** key[], const int keySize)  
Врши проверу **CRC32** функцијом, да ли је дошло до нежељених измена у шифрованој датотеци in.
- int **CypherAES** (**UChar** \*\*inFileName, const **UChar** keyIn[], const int keySize)  
Шифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.
- int **DecypherAES** (**UChar** \*\*inFileName, const **UChar** keyIn[], const int keySize)  
Дешифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.
- int **main** ()

## 2.1.1 Документација функције

### 2.1.1.1 void AddRoundKey ( UChar state[4], const UInt32 key[] )

Врши операцију еклузивне дисјункције блока бајтова величине **STATE\_SIZE** и кључа.

Добијени резултат се смешта у променљиву state.

Параметри

state	[in/out] Матрица 4 * 4 бајтова који се XOR-ује са кључем.
key	[in] Део кључа са којим се XOR-ује.  <b>UChar</b> state[4][4] = {0}; <b>UInt32</b> key[4] = {0}; <b>AddRoundKey</b> (state, key); // XOR-ује state са кључем чије су све вредности 0

Дефиниција у линији 17 датотеке aes.c.

### 2.1.1.2 UInt32 CRC32 ( UInt32 crc, const UChar state[] )

Враћа вредност CRC-а датог блока state од **STATE\_SIZE** бајтова и crc.

## Параметри

crc	[in] Вредност са којом се ради <a href="#">CRC32</a>
state	[in] Блок бајтова који се CRC-ује.

## Враћа

Новодобијена вредност CRC функције након примене CRC-a на crc и state

```
UInt32 crc = 0;
UChar state[STATE_SIZE] = {0};
crc = CRC32(crc, state); // crc = 0xecbb4b55
```

Дефиниција у линији 627 датотеке aes.c.

2.1.1.3 int CRCCheck ( FILE \* in, const UInt32 key[], const int keySize )

Врши проверу [CRC32](#) функцијом, да ли је дошло до нежељених измена у шифрованој датотеци in.

## Параметри

in	[in/out] Отворена улазна датотеке који је отворен за читање бинарних датоека код којег желимо да проверимо да ли је дошло до нежељених измена. Почетна позиција за читање мора бити на почетку датотке.
key	[in] Развијени облик кључа којим је шифрована датотека in.
keySize	[in] Величина кључа којим је шифрована датотека in.

## Враћа

0 ако није дошло до нежељених измена у шифрованој датотеци in.

1 ако је дошло до нежељених измена у шифрованој датотеци in.

```
FILE *in = fopen("ulaz.in.crypt", "rb");
UChar keyIn[] = "\x2b\xe\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
UInt32 *key = NULL;
KeyExpansion(keyIn, &key, strlen(keyIn) * 8);
if (CRCCheck(in, key, strlen(keyIn) * 8) == 0)
    printf("Dobro sifrovana");
else printf("Lose");
fclose(in);
```

## Види

[KeyExpansion](#)

Дефиниција у линији 639 датотеке aes.c.

2.1.1.4 int CypherAES ( UChar \*\* inFileName, const UChar keyIn[], const int keySize )

Шифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.

Име датотеке, \*inFileName мора бити алоцирано на heap-у. Кључ keyIn којим се шифрује мора бити величине 128, 192 или 256 битова. Величина кључа се прослеђује у keySize. Име датотеке која се добија шифровањем се прослеђује у \*inFileName. Датотека која се шифрује мора бити мања од 4GB.

## Параметри

inFileName	[in/out] Име датотеке која се шифрује и име под којим је шифрована.
keyIn	[in] Кључ којим се шифрује датотека
keySize	[in] Величина кључа којим се шифрује датотека.

Враћа

- 0 ако је успешно шифрована датотека
- 1 ако је дошло до грешке при читању из датотеке (непостојећа датотека, грешке у систему).
- 2 ако је дошло до грешке при писању у датотеку.
- 3 ако нема довољно меморије.
- 4 ако дужина имена није добра (већа од максималне дозвољене у WINDOWS-у).
- 5 ако дужина кључа није добра.

```
UChar *inputName, key[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
inputName = (UChar*)malloc(256);
strcpy(inputName, "ulaz.in");
CypherAES(&inputName, key, strlen(key) * 8); // шифрује датотеку ulaz.in pomocu kljuka key
```

Упозорење

Величина датотеке која се шифрује мора бити мања од 4GB.

Дефиниција у линији 706 датотеке aes.c.

2.1.1.5 UChar\* CypherFileName ( const UChar \* inFileName )

Прави име шифроване датотеке од имена датотеке који треба да шифрује

Додаје `CRYPT_TEXT` на крај имена шифроване датотеке inFileName. Ако је име путање које би се добило веће од максималне дужине имена путање коју WINDOWS може да прими (`MAX_FILE_PATH`), на крај имена додаје `CRYPT_TEXT` преко онолико карактера колико треба да се дода `CRYPT_TEXT`, а да не пређе ограничење у WINDOWS.

Параметри

inFileName	[in] Име улазне датотеке који се шифрује.
------------	---

Враћа

NULL ако нема довољно меморије.  
Иначе враћа име шифроване датотеке.

```
char *t = "tata";
t = CypherFileName(t); // t postaje tata.crypt;
```

Дефиниција у линији 568 датотеке aes.c.

2.1.1.6 void DecryptState ( const UChar in[], UChar out[], const UInt32 key[], const int keySize )

Дешифрује АЕС-ом низ бајтова величине `STATE_SIZE` задат са in помоћу низа кључева key (добијених `KeyExpansion`) и уписује у out.

in Мора бити величине бар `STATE_SIZE`, out мора имати алоцирану меморију изван функције.

Параметри

in	[in] Низ бајтова величине <code>STATE_SIZE</code> који се дешифрује АЕС-ом.
out	[out] Низ дешифрованих бајтова величине <code>STATE_SIZE</code>
key	[in] Низ експандованих кључева.
keySize	[in] Величина кључа којим се дешифрује.  <pre>UChar out[STATE_SIZE], in[STATE_SIZE] = {0}, keyIn[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c"; UInt32 *key; KeyExpansion(keyIn, &amp;key, 128); DecryptState(in, out, key, 128); // out је низ дешифрованих бајтова in кључем key.</pre>

## Упозорење

Простор од `STATE_SIZE` за out мора бити алоциран изван функције.

Дефиниција у линији 441 датотеке aes.c.

2.1.1.7 int DecypherAES ( UChar \*\* inFileName, const UChar keyIn[], const int keySize )

Дешифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.

Име датотеке, \*inFileName мора бити алоцирано на heap-у. Кључ keyIn којим се дешифрује мора бити величине 128, 192 или 256 бита. Величина кључа се прослеђује у keySize. Име датотеке која се добија дешифровањем се прослеђује у \*inFileName. Датотека која се дешифрује мора бити мања од 4GB (не укључујући заглавље).

## Параметри

inFileName	[in/out] Име датотеке која се дешифрује и име под којим је дешифрована.
keyIn	[in] Кључ којим се дешифрује датотека
keySize	[in] Величина кључа којим се дешифрује датотека.

## Враћа

- 0 ако је успешно дешифрована датотека
- 1 ако је дошло до грешке при читању из датотеке (непостојећа датотека, грешке у систему).
- 2 ако је дошло до грешке при писању у датотеку.
- 3 ако нема довољно меморије.
- 4 ако дужина имена није добра (већа од максималне дозвољене у WINDOWS-у).
- 5 ако дужина кључа није добра.
- 6 ако није добро шифрована датотека уз помоћ кључа keyIn

```
UChar *inputName, key[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
inputName = (UChar*)malloc(256);
strcpy(inputName, "ulaz.in.crypt");
DecypherAES(&inputName, key, strlen(key) * 8); // датотека ulaz.in.crypt се дешифрује помоћу
kljuka key
```

## Упозорење

Величина датотеке која се шифрује мора бити мања од 4GB (искључујући заглавља).

Дефиниција у линији 795 датотеке aes.c.

2.1.1.8 int DeCypherFileName ( UChar \*\* inFileName )

Враћа име датотеке које ће имати након што се дешифрује.

Проверава да ли постоји датотека са истим именом као \*inFileName. Ако постоји додаје насумичан број на почетак док дато име не буде постојало. Кад дужина путање до имена фајла са додатим насумичним бројем пређе дужину `MAX_FILE_PATH` додаје се број тако да се крај имена губи. \*inFileName мора бити алоциран динамички.

## Параметри

inFileName	[in/out] Име датотеке који се дешифрује.
------------	--

## Враћа

- 0 ако је све добро протекло.
- 1 ако нема довољно меморије.
- 2 ако је путања неисправна (предугачко име прослеђено).

```
UChar *t = (UChar*)malloc(5);
t[0] = '\0';
strcat(t, "tata");
DeCypherFileName(&t);
```

Дефиниција у линији 592 датотеке aes.c.

2.1.1.9 void EncryptState ( const UChar in[], UChar out[], const UInt32 key[], const int keySize )

Шифрује АЕС-ом низ бајтова величине `STATE_SIZE` задат са in помоћу низа кључева key (добijених `KeyExpansion`) и уписује у out.

in Мора бити величине бар `STATE_SIZE`, out мора имати алоцирану меморију изван функције.

## Параметри

in	[in] Низ бајтова величине <code>STATE_SIZE</code> који се шифрује АЕС-ом.
out	[out] Низ шифрованих бајтова величине <code>STATE_SIZE</code>
key	[in] Низ експандованих кључева.
keySize	[in] Величина кључа којим се шифрује.  <pre>UChar out[STATE_SIZE], in[STATE_SIZE] = {0}, keyIn[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c"; UInt32 *key; KeyExpansion(keyIn, &amp;key, 128); EncryptState(in, out, key, 128); // out је низ шифрованих бајтова in кључем key.</pre>

## Упозорење

Простор од `STATE_SIZE` за out мора бити алоциран изван функције.

Дефиниција у линији 374 датотеке aes.c.

2.1.1.10 void InvMixColumns ( UChar state[][4] )

Инверзно миксује колоне матрице бајтова state.

Инверзно миксовање се врши множењем са одговарајућом матрицом у Галоаовом пољу. Крајњи резултат је матрица која је првобитно била миксована.

## Параметри

state	[in/out] Матрица бајтова у којој се вршим миксовање колона.  <pre>UChar state[4][4] = { { 2 }, { 7 }, { 0 }, { 5 } }; InvMixColumns(state); // state = {{0}, {1}, {2}, {3}}, матрица која се добија миксовањем колона.</pre>
-------	--

Дефиниција у линији 235 датотеке aes.c.

2.1.1.11 void InvShiftRows ( UChar state[][4] )

Ротира бајтове у врстама, у i-тој врсти за i места удесно.

Матрица бајтова state 4 \* 4 трансформише се у нову тако да се свака i-та врста ротира удесно i места.

Параметри

state	[in/out] Матрица бајтова у којој се бајтови по врстама ротирају.  Uchar state[4][4] = {{0}, {1}, {2}, {3}}; ShiftRows(state); // state[4][4] = {{0}, {0, 1, 0, 0}, {0, 0, 2, 0}, {0, 0, 0, 3}}
-------	---

Дефиниција у линији 121 датотеке aes.c.

2.1.1.12 void InvSubBytes ( UChar state[4][4] )

Мења вредност сваког појединачног бајта из state одговарајућом вредношћу из INVS\_BOX.

Параметри

state	[in/out] Матрица 4 *4 бајтова над чијим вредностима се врши замена.  UChar state[4][4] = {0}; SubBytes(state); Замењује сваки бајт из state са одговарајућом вредношћу из INVS_BOX
-------	---

Дефиниција у линији 79 датотеке aes.c.

2.1.1.13 int KeyExpansion ( const UChar key[], UInt32 \*\* w, const int keySize )

Развија кључ key у низ 4-бајтних кључева w.

Добијених 4-бајтних кључева има колико и (рунди шифровања + 1) \* 4. Рунди шифровања има у зависности од величине кључа. Функција враћа вредност int која указује на тип грешке (ако га има).

Параметри

key	[in] Кључ који се развија.
keySize	[in] Величина кључа који се развија.
w	[out] Показивач на низ експандованих кључева од кључа key

Враћа

-1 ако нема довољно меморије за експанзију кључа.

1 ако је прослеђена лоша дужина кључа.

0 ако је све протекло у реду.

```
UChar keyIn[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
UInt32 *key;
KeyExpansion(keyIn, &key, 128); // Развија 128-битни keyIn кључ keyIn.
```

Дефиниција у линији 341 датотеке aes.c.

2.1.1.14 int main ( )

Дефиниција у линији 886 датотеке aes.c.

2.1.1.15 void MixColumns ( UChar state[4][4] )

Миксује колоне матрице бајтова state.

Миксовање се врши множењем са одговарајућом матрицом у Галоаовом пољу.

## Параметри

state	[in/out] Матрица бајтова у којој се вршим миксовање колона.  Uchar state[4][4] = {{0}, {1}, {2}, {3}}; MixColumns(state); // state = {{2}, {7}, {0}, {5}}, матрица која се добија миксовањем колона.
-------	---

Дефиниција у линији 144 датотеке aes.c.

2.1.1.16 int ReadData ( FILE \* f, UChar state[], UInt32 \* sizeToEnd )

Чита из датотеке f блок бајтова STATE\_SIZE и смешта у state.

Ако је број преосталих бајтова до краја датотеке (sizeToEnd) мањи од STATE\_SIZE, прочитаће само толико бајтова и уписати их у state. Остатак се попуњава нулама. sizeToEnd је показивач на величину бајтова колико се чита. Његов број смањује се за STATE\_SIZE или ако је мањи од STATE\_SIZE поставља се на 0. Низ state мора имати бар 16 бајтова меморије.

## Параметри

f	[in/out] Показивач на датотеку из које се бајтови читају.
state	[out] Низ бајтова (UChar) у који се смештају прочитани бајтови.
sizeToEnd	[in/out] Величина којом се одређује колико се бајтова чита (и чија вредност се умањује за број прочитаних бајтова).

## Враћа

- 1 ако је све добро протекло.
- 0 ако је дошло до грешке при читању датотеке.

```
FILE *in = fopen("Ulaz.in", "r");
UChar state[STATE_SIZE];
UInt32 sizeToEnd = 10;
if (!ReadData(in, state, &sizeToEnd))
    printf("Greska pri citanju\n");
```

Дефиниција у линији 524 датотеке aes.c.

2.1.1.17 void RotWord ( UInt32 \* wordPt )

1-бајтна кружна ротација 4-бајтне(UInt32) речи улево.

## Параметри

wordPt	[in/out] Показивач на UInt32 реч која се ротира.  UInt32 t= 0x01; RotWor(&t); // t = 0x0100;
--------	---

Дефиниција у линији 323 датотеке aes.c.

2.1.1.18 void ShiftRows ( UChar state[][4] )

Ротира бајтове у врстама, у i-тој врсти за i места улево.

Матрица бајтова state 4 \* 4 трансформише се у нову тако да се свака i-та врста ротира улево i места.



## Параметри

state	[in/out] Матрица бајтова у којој се бајтови по врстама ротирају.  Uchar state[4][4] = {{0}, {1}, {2}, {3}} ShiftRows(state); // state[4][4] = {{0}, {0, 0, 0, 1}, {0, 0, 2, 0}, {0, 3, 0, 0}}
-------	--

Дефиниција у линији 98 датотеке aes.c.

## 2.1.1.19 UInt32 SizeOfFile ( const UChar \* name )

Враћа величину датотеке имена name која се налази у истом фолдеру у којем је програм.

## Параметри

name	[in] Име датотеке чија се величина тражи.
------	---

## Враћа

Величина датотеке у бајтовима.

```
UChar *name = "Ulaz.in";
UInt32 size = SizeOfFile(name); // vraca velicinu datoteke sa imenom name
```

## Упозорење

Величина датотеке не сме бити већа од 4GB. Ако дође до грешке при читању датотеке, непознат је резултат.

Дефиниција у линији 506 датотеке aes.c.

## 2.1.1.20 UInt32 StateToUInt32 ( const UChar state[] )

Преводи низ од 16 бајтова у број од 4 бајта (UInt32).

Бајтови са нижим индексима у низу иду у више бајтове броја (они који имају већу тежину у броју). Најнижи бајт броја је у state[STATE\_SIZE-1]. Највиши бајт броја је у state[STATE\_SIZE-4].

## Параметри

state	[in/out] Низ бајтова у који се пребацује број.
-------	--

## Враћа

4-бајтни број који се добија из state.

```
UInt32 t;
UChar state[STATE_SIZE] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1};
t = StateToUInt32(state) // t = 1
```

## Напомена

Функција је неопходна јер се не зна да ли рачунар на ком се покреће програм има LITTLE-↵ ENDIAN или BIG-ENDIAN.

Дефиниција у линији 557 датотеке aes.c.

## 2.1.1.21 void SubBytes ( UChar state[][4] )

Мења вредност сваког појединачног бајта из state одговарајућом вредношћу из S\_BOX.

## Параметри

state	[in/out] Матрица 4 *4 бајтова над чијим вредностима се врши замена.  <code>UChar state[4][4] = {0};</code> <code>SubBytes(state);</code> Заменује сваки бајт из state са одговарајућом вредношћу из <code>S_BOX</code>
-------	---

Дефиниција у линији 60 датотеке aes.c.

2.1.1.22 UInt32 SubWord ( const UInt32 word )

Замена сваког бајта 4-бајтне речи word одговарајућим бајтом из `S_BOX`.

## Параметри

word	[in] 4-бајтна реч типа <code>UInt32</code> код које се мењају бајтови.
------	--

## Враћа

4-бајтна реч која се добија заменом бајтова из word

```
UInt32 t = 0x1;
t = SubWord(t); // t = 0x6363637c
```

Дефиниција у линији 327 датотеке aes.c.

2.1.1.23 UInt32 UCharToUInt32 ( const UChar byte[] )

Претвара низ 4 бајта у `UInt32`.

## Параметри

byte	[in] Низ 4 бајта (тип <code>UChar</code> ).
------	---

## Враћа

Враћа број типа `UInt32` чија је представа byte

```
UChar byte[4] = { \x0, \x1, \x2, \x3 }
UInt32 t = UCharToUInt32(byte); // t добија вредност 0x0123
```

Дефиниција у линији 8 датотеке aes.c.

2.1.1.24 void UInt32ToState ( const UInt32 num, UChar state[] )

Преводи 4-бајтни број типа `UInt32` у низ од 16 бајтова.

Виши бајтови броја (они који имају већу тежину у броју) иду на ниже индексе у низу. Остали бајтови се попуњавају са нулама. Низ мора имати меморију од бар `STATE_SIZE` бајтова. Најнижи бајт броја се ставља у state[STATE\_SIZE-1]

## Параметри

num	[in] Број који пребацујемо у низ бајтова.
state	[in/out] Низ бајтова у који се пребацује број.  <code>UInt32 t = 1;</code> <code>UChar state[STATE_SIZE];</code> <code>UInt32ToState(t, state);</code> // последњи бајт state је 1, сви остали су 0

## Напомена

Функција је неопходна јер се не зна да ли рачунар на ком се покреће програм има LITTLE-↵  
ENDIAN или BIG-ENDIAN.

Дефиниција у линији 549 датотеке aes.c.

2.1.1.25 int WriteData ( FILE \* f, const UChar state[] )

Пише у датотеку f блок бајтова STATE\_SIZE из state.

Параметри

f	[in/out] Показивач на датотеку у који се бајтови пишу.
state	[in] Низ бајтова (UChar) који се уписују.

## Враћа

1 ако је све добро протекло.  
0 ако је дошло до грешке при писању датотеке.

```
FILE *out = fopen("Izlaz.out", "w");
UChar state[STATE_SIZE] = {0};
if (!WriteData(out, state)) // pise u datoteku Izlaz.out STATE_SIZE bajtova ciji su svi bitovi 0
printf("Greska pri pisanju\n");
```

Дефиниција у линији 542 датотеке aes.c.

## 2.2 AES/aes.h Референца датотеке

Фајл садржи функције за шифровање и дешифровање датотека АЕС-ом.

```
#include <stdio.h>
```

## Дефиниције

- #define \_CRT\_SECURE\_NO\_DEPRECATED
- #define \_aes\_h\_
- #define NUM\_ROW 4
- #define STATE\_SIZE 16
- #define CRYPT\_TEXT ".crypt"
- #define DIGIT\_NUM 10
- #define MAX\_FILE\_PATH 260

## Дефиниције типова

- typedef unsigned \_\_int32 UInt32
- typedef unsigned \_\_int8 UChar

## Функције

- UInt32 UCharToUInt32 (const UChar byte[])  
Претвара низ 4 бајта у UInt32.
- void AddRoundKey (UChar state[4], const UInt32 key[])  
Врши операцију еклузивне дисјункције блока бајтова величине STATE\_SIZE и кључа.

- void **SubBytes** (**UChar** state[4])  
 Мења вредност сваког појединачног бајта из state одговарајућом вредношћу из **S\_BOX**.
- void **InvSubBytes** (**UChar** state[4])  
 Мења вредност сваког појединачног бајта из state одговарајућом вредношћу из **INVS\_BOX**.
- void **ShiftRows** (**UChar** state[4])  
 Ротира бајтове у врстама, у i-тој врсти за i места улево.
- void **InvShiftRows** (**UChar** state[4])  
 Ротира бајтове у врстама, у i-тој врсти за i места удесно.
- void **MixColumns** (**UChar** state[4])  
 Миксује колоне матрице бајтова state.
- void **InvMixColumns** (**UChar** state[4])  
 Инверзно миксује колоне матрице бајтова state.
- void **RotWord** (**UInt32** \*wordPt)  
 1-бајтна кружна ротација 4-бајтне(**UInt32**) речи улево.
- **UInt32** **SubWord** (const **UInt32** word)  
 Замена сваког бајта 4-бајтне речи word одговарајућим бајтом из **S\_BOX**.
- int **KeyExpansion** (const **UChar** key[], **UInt32** \*\*w, const int keySize)  
 Развија кључ key у низ 4-бајтних кључева w.
- void **EncryptState** (const **UChar** in[], **UChar** out[], const **UInt32** key[], const int keySize)  
 Шифрује АЕС-ом низ бајтова величине **STATE\_SIZE** задат са in помоћу низа кључева key (добijenих **KeyExpansion**) и уписује у out.
- void **DecryptState** (const **UChar** in[], **UChar** out[], const **UInt32** key[], const int keySize)  
 Дешифрује АЕС-ом низ бајтова величине **STATE\_SIZE** задат са in помоћу низа кључева key (добijenих **KeyExpansion**) и уписује у out.
- **UInt32** **SizeOfFile** (const **UChar** \*name)  
 Враћа величину датотеке имена name која се налази у истом фолдеру у којем је програм.
- int **ReadData** (FILE \*f, **UChar** state[], **UInt32** \*sizeToEnd)  
 Чита из датотеке f блок бајтова **STATE\_SIZE** и смешта у state.
- int **WriteData** (FILE \*f, const **UChar** state[])  
 Пише у датотеку f блок бајтова **STATE\_SIZE** из state.
- void **UInt32ToState** (const **UInt32** num, **UChar** state[])  
 Преводи 4-бајтни број типа **UInt32** у низ од 16 бајтова.
- **UInt32** **StateToUInt32** (const **UChar** state[])  
 Преводи низ од 16 бајтова у број од 4 бајта (**UInt32**).
- **UChar** \* **CypherFileName** (const **UChar** \*inFileName)  
 Прави име шифроване датотеке од имена датотеке који треба да шифрује
- int **DeCypherFileName** (**UChar** \*\*inFileName)  
 Враћа име датотеке које ће имати након што се дешифрује.
- **UInt32** **CRC32** (**UInt32** crc, const **UChar** state[])  
 Враћа вредност CRC-а датог блока state од **STATE\_SIZE** бајтова и crc.
- int **CRCCheck** (FILE \*in, const **UInt32** key[], const int keySize)  
 Врши проверу **CRC32** функцијом, да ли је дошло до нежељених измена у шифрованој датотеци in.
- int **CypherAES** (**UChar** \*\*inFileName, const **UChar** keyIn[], const int keySize)  
 Шифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.
- int **DecypherAES** (**UChar** \*\*inFileName, const **UChar** keyIn[], const int keySize)  
 Дешифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.

## Променљиве

- const [UChar](#) [S\\_BOX](#) [16][16]
- const [UChar](#) [INVS\\_BOX](#) [16][16]
- const [UChar](#) [G\\_FIELD\\_MUL](#) [256][6]
- const [UInt32](#) [ROUND\\_CONSTANT](#) []
- const [UInt32](#) [CRC32\\_TABLE](#) []

### 2.2.1 Опширније

Фајл садржи функције за шифровање и дешифровање датотека АЕС-ом.

Аутор

Ђорђе Живановић

Дефиниција у датотеци [aes.h](#).

### 2.2.2 Документација дефиниције

#### 2.2.2.1 #define \_aes\_h\_

Дефиниција у линији 8 датотеке [aes.h](#).

#### 2.2.2.2 #define \_CRT\_SECURE\_NO\_DEPRECATED

Дефиниција у линији 6 датотеке [aes.h](#).

#### 2.2.2.3 #define CRYPT\_TEXT ".crypt"

Екстензија криптоване датотеке.

Дефиниција у линији 14 датотеке [aes.h](#).

#### 2.2.2.4 #define DIGIT\_NUM 10

Број цифара у декадном систему.

Дефиниција у линији 15 датотеке [aes.h](#).

#### 2.2.2.5 #define MAX\_FILE\_PATH 260

Максимална дужина путање у WINDOWS

Дефиниција у линији 16 датотеке [aes.h](#).

#### 2.2.2.6 #define NUM\_ROW 4

Број врста у једном блоку.

Дефиниција у линији 12 датотеке [aes.h](#).

#### 2.2.2.7 #define STATE\_SIZE 16

Број бајтова у једном блоку

Дефиниција у линији 13 датотеке [aes.h](#).

## 2.2.3 Документација дефиниције типа

### 2.2.3.1 typedef unsigned \_\_int8 UChar

Ако unsigned int није 4 бита.

Дефиниција у линији 11 датотеке aes.h.

### 2.2.3.2 typedef unsigned \_\_int32 UInt32

Дефиниција у линији 10 датотеке aes.h.

## 2.2.4 Документација функције

### 2.2.4.1 void AddRoundKey ( UChar state[16], const UInt32 key[4] )

Врши операцију еклузивне дисјункције блока бајтова величине `STATE_SIZE` и кључа.

Добијени резултат се смешта у променљиву state.

Параметри

state	[in/out] Матрица 4 * 4 бајтова који се XOR-ује са кључем.
key	[in] Део кључа са којим се XOR-ује.  <pre>UChar state[4][4] = {0}; UInt32 key[4] = {0}; AddRoundKey(state, key); // XOR-ује state са кључем чије су све вредности 0</pre>

Дефиниција у линији 17 датотеке aes.c.

### 2.2.4.2 UInt32 CRC32 ( UInt32 crc, const UChar state[16] )

Враћа вредност CRC-а датог блока state од `STATE_SIZE` бајтова и crc.

Параметри

crc	[in] Вредност са којом се ради <code>CRC32</code>
state	[in] Блок бајтова који се CRC-ује.

Враћа

Новодобијена вредност CRC функције након примене CRC-а на crc и state

```
UInt32 crc = 0;
UChar state[STATE_SIZE] = {0};
crc = CRC32(crc, state); // crc = 0xecbb4b55
```

Дефиниција у линији 627 датотеке aes.c.

### 2.2.4.3 int CRCCheck ( FILE \* in, const UInt32 key[4], const int keySize )

Врши проверу `CRC32` функцијом, да ли је дошло до нежељених измена у шифрованој датотеци in.

Параметри

in	[in/out] Отворена улазна датотеке који је отворен за читање бинарних датоека код којег желимо да проверимо да ли је дошло до нежељених измена. Почетна позиција за читање мора бити на почетку датотке.
key	[in] Развијени облик кључа којим је шифрована датотека in.
keySize	[in] Величина кључа којим је шифрована датотека in.

Враћа

0 ако није дошло до нежељених измена у шифрованој датотеци in.

1 ако је дошло до нежељених измена у шифрованој датотеци in.

```
FILE *in = fopen("ulaz.in.crypt", "rb");
UChar keyIn[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
UInt32 *key = NULL;
KeyExpansion(keyIn, &key, strlen(keyIn) * 8);
if (CRCCheck(in, key, strlen(keyIn) * 8) == 0)
    printf("Dobro sifrovana");
else printf("Lose");
fclose(in);
```

Види

[KeyExpansion](#)

Дефиниција у линији 639 датотеке aes.c.

2.2.4.4 int CypherAES ( UChar \*\* inFileName, const UChar keyIn[], const int keySize )

Шифрује датотеку са именом \*inFileName помоћу кључа keyIn, величине keySize.

Име датотеке, \*inFileName мора бити алоцирано на heap-у. Кључ keyIn којим се шифрује мора бити величине 128, 192 или 256 битова. Величина кључа се прослеђује у keySize. Име датотеке која се добија шифровањем се прослеђује у \*inFileName. Датотека која се шифрује мора бити мања од 4GB.

Параметри

inFileName	[in/out] Име датотеке која се шифрује и име под којим је шифрована.
keyIn	[in] Кључ којим се шифрује датотека
keySize	[in] Величина кључа којим се шифрује датотека.

Враћа

0 ако је успешно шифрована датотека

1 ако је дошло до грешке при читању из датотеке (непостојећа датотека, грешке у систему).

2 ако је дошло до грешке при писању у датотеку.

3 ако нема довољно меморије.

4 ако дужина имена није добра (већа од максималне дозвољене у WINDOWS-у).

5 ако дужина кључа није добра.

```
UChar *inputName, key[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
inputName = (UChar*)malloc(256);
strcpy(inputName, "ulaz.in");
CypherAES(&inputName, key, strlen(key) * 8); // sifruje datoteku ulaz.in pomocu kljuka key
```

Упозорење

Величина датотеке која се шифрује мора бити мања од 4GB.

Дефиниција у линији 706 датотеке aes.c.

#### 2.2.4.5 UChar\* CypherFileName ( const UChar \* inFileName )

Прави име шифроване датотеке од имена датотеке који треба да шифрује

Додаје `CRYPT_TEXT` на крај имена шифроване датотеке `inFileName`. Ако је име путање које би се добило веће од максималне дужине имена путање коју WINDOWS може да прими (`MAX_FILE_PATH`), на крај имена додаје `CRYPT_TEXT` преко онолико карактера колико треба да се дода `CRYPT_TEXT`, а да не пређе ограничење у WINDOWS.

Параметри

<code>inFileName</code>	[in] Име улазне датотеке који се шифрује.
-------------------------	---

Враћа

NULL ако нема довољно меморије.  
Иначе враћа име шифроване датотеке.

```
char *t = "tata";
t = CypherFileName(t); // t postaje tata.crypt;
```

Дефиниција у линији 568 датотеке `aes.c`.

#### 2.2.4.6 void DecryptState ( const UChar in[], UChar out[], const UInt32 key[], const int keySize )

Дешифрује АЕС-ом низ бајтова величине `STATE_SIZE` задат са `in` помоћу низа кључева `key` (добијених `KeyExpansion`) и уписује у `out`.

`in` Мора бити величине бар `STATE_SIZE`, `out` мора имати алоцирану меморију изван функције.

Параметри

<code>in</code>	[in] Низ бајтова величине <code>STATE_SIZE</code> који се дешифрује АЕС-ом.
<code>out</code>	[out] Низ дешифрованих бајтова величине <code>STATE_SIZE</code>
<code>key</code>	[in] Низ експандованих кључева.
<code>keySize</code>	[in] Величина кључа којим се дешифрује.  <pre>UChar out[STATE_SIZE], in[STATE_SIZE] = {0}; keyIn[] = "\x2b\xe\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c"; UInt32 *key; KeyExpansion(keyIn, &amp;key, 128); DecryptState(in, out, key, 128); // out је низ дешифрованих бајтова in кључем key.</pre>

Упозорење

Простор од `STATE_SIZE` за `out` мора бити алоциран изван функције.

Дефиниција у линији 441 датотеке `aes.c`.

#### 2.2.4.7 int DecypherAES ( UChar \*\* inFileName, const UChar keyIn[], const int keySize )

Дешифрује датотеку са именом `*inFileName` помоћу кључа `keyIn`, величине `keySize`.

Име датотеке, `*inFileName` мора бити алоцирано на heap-у. Кључ `keyIn` којим се дешифрује мора бити величине 128, 192 или 256 битава. Величина кључа се прослеђује у `keySize`. Име датотеке која се добија дешифровањем се прослеђује у `*inFileName`. Датотека која се дешифрује мора бити мања од 4GB (не укључујући заглавље).



## Параметри

inFileName	[in/out] Име датотеке која се дешифрује и име под којим је дешифрована.
keyIn	[in] Кључ којим се дешифрује датотека
keySize	[in] Величина кључа којим се дешифрује датотека.

## Враћа

- 0 ако је успешно дешифрована датотека
- 1 ако је дошло до грешке при читању из датотеке (непостојећа датотека, грешке у систему).
- 2 ако је дошло до грешке при писању у датотеку.
- 3 ако нема довољно меморије.
- 4 ако дужина имена није добра (већа од максималне дозвољене у WINDOWS-у).
- 5 ако дужина кључа није добра.
- 6 ако није добро шифрована датотека уз помоћ кључа keyIn

```
UChar *inputName, key[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
inputName = (UChar*)malloc(256);
strcpy(inputName, "ulaz.in.crypt");
DecypherAES(&inputName, key, strlen(key) * 8); // datoteka ulaz.in.crypt se desifruje pomocu
kljuca key
```

## Упозорење

Величина датотеке која се шифрује мора бити мања од 4GB (искључујући заглавља).

Дефиниција у линији 795 датотеке aes.c.

2.2.4.8 int DeCypherFileName ( UChar \*\* inFileName )

Враћа име датотеке које ће имати након што се дешифрује.

Проверава да ли постоји датотека са истим именом као \*inFileName. Ако постоји додаје насумичан број на почетак док дато име не буде постојало. Кад дужина путање до имена фајла са додатим насумичним бројем пређе дужину `MAX_FILE_PATH` додаје се број тако да се крај имена губи. \*inFileName мора бити алоциран динамички.

## Параметри

inFileName	[in/out] Име датотеке који се дешифрује.
------------	--

## Враћа

- 0 ако је све добро протекло.
- 1 ако нема довољно меморије.
- 2 ако је путања неисправна (предугачко име прослеђено).

```
UChar *t = (UChar*)malloc(5);
t[0] = '\0';
strcat(t, "tata");
DeCypherFileName(&t);
```

Дефиниција у линији 592 датотеке aes.c.

2.2.4.9 void EncryptState ( const UChar in[], UChar out[], const UInt32 key[], const int keySize )

Шифрује АЕС-ом низ бајтова величине `STATE_SIZE` задат са in помоћу низа кључева key (добijenih [KeyExpansion](#)) и уписује у out.

in Мора бити величине бар `STATE_SIZE`, out мора имати алоцирану меморију изван функције.

## Параметри

in	[in] Низ бајтова величине <code>STATE_SIZE</code> који се шифрује АЕС-ом.
out	[out] Низ шифрованих бајтова величине <code>STATE_SIZE</code>
key	[in] Низ експандованих кључева.
keySize	[in] Величина кључа којим се шифрује.  <pre>UChar out[STATE_SIZE], in[STATE_SIZE] = {0}, keyIn[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c"; UInt32 *key; KeyExpansion(keyIn, &amp;key, 128); EncryptState(in, out, key, 128); // out је низ шифрованих бајтова in кључем key.</pre>

## Упозорење

Простор од `STATE_SIZE` за out мора бити алоциран изван функције.

Дефиниција у линији 374 датотеке aes.c.

2.2.4.10 void InvMixColumns ( UChar state[4] )

Инверзно миксује колоне матрице бајтова state.

Инверзно миксовање се врши множењем са одговарајућом матрицом у Галоаовом пољу. Крајњи резултат је матрица која је првобитно била миксована.

## Параметри

state	[in/out] Матрица бајтова у којој се вршим миксовање колона.  <pre>UChar state[4][4] = { { 2 }, { 7 }, { 0 }, { 5 } }; InvMixColumns(state); // state = {{0}, {1}, {2}, {3}}, матрица која се добија миксовањем колона.</pre>
-------	--

Дефиниција у линији 235 датотеке aes.c.

2.2.4.11 void InvShiftRows ( UChar state[4] )

Ротира бајтове у врстама, у i-тој врсти за i места удесно.

Матрица бајтова state 4 \* 4 трансформише се у нову тако да се свака i-та врста ротира удесно i места.

## Параметри

state	[in/out] Матрица бајтова у којој се бајтови по врстама ротирају.  <pre>UChar state[4][4] = {{0}, {1}, {2}, {3}} ShiftRows(state); // state[4][4] = {{0}, {0, 1, 0, 0}, {0, 0, 2, 0}, {0, 0, 0, 3}}</pre>
-------	--

Дефиниција у линији 121 датотеке aes.c.

2.2.4.12 void InvSubBytes ( UChar state[4] )

Мења вредност сваког појединачног бајта из state одговарајућом вредношћу из `INVS_BOX`.

## Параметри

state	[in/out] Матрица 4 *4 бајтова над чијим вредностима се врши замена.  UChar state[4][4] = {0}; SubBytes(state); Заменjuje сваки бајт из state са одговарајућом вредношћу из INVS_BOX
-------	--

Дефиниција у линији 79 датотеке aes.c.

2.2.4.13 int KeyExpansion ( const UChar key[], UInt32 \*\* w, const int keySize )

Развија кључ key у низ 4-бајтних кључева w.

Добијених 4-бајтних кључева има колико и (рунди шифровања + 1) \* 4. Рунди шифровања има у зависности од величине кључа. Функција враћа вредност int која указује на тип грешке (ако га има).

Параметри

key	[in] Кључ који се развија.
keySize	[in] Величина кључа који се развија.
w	[out] Показивач на низ експандованих кључева од кључа key

Враћа

- 1 ако нема довољно меморије за експанзију кључа.
- 1 ако је прослеђена лоша дужина кључа.
- 0 ако је све протекло у реду.

```
UChar keyIn[] = "\x2b\x7e\x15\x16\x28\xae\xd2\xa6\xab\xf7\x15\x88\x09\xcf\x4f\x3c";
UInt32 *key;
KeyExpansion(keyIn, &key, 128); // Развија 128-битни keyIn кључ keyIn.
```

Дефиниција у линији 341 датотеке aes.c.

2.2.4.14 void MixColumns ( UChar state[][4] )

Миксује колоне матрице бајтова state.

Миксовање се врши множењем са одговарајућом матрицом у Галоаовом пољу.

Параметри

state	[in/out] Матрица бајтова у којој се вршим миксовање колона.  UChar state[4][4] = {{0}, {1}, {2}, {3}}; MixColumns(state); // state = {{2}, {7}, {0}, {5}}, матрица која се добија миксовањем колона.
-------	---

Дефиниција у линији 144 датотеке aes.c.

2.2.4.15 int ReadData ( FILE \* f, UChar state[], UInt32 \* sizeToEnd )

Чита из датотеке f блок бајтова STATE\_SIZE и смешта у state.

Ако је број преосталих бајтова до краја датотеке (sizeToEnd) мањи од STATE\_SIZE, прочитаће само толико бајтова и уписати их у state. Остатак се попуњава нулама. sizeToEnd је показивач на величину бајтова колико се чита. Његов број смањује се за STATE\_SIZE или ако је мањи од STATE\_SIZE поставља се на 0. Низ state мора имати бар 16 бајтова меморије.

## Параметри

f	[in/out] Показивач на датотеку из које се бајтови читају.
state	[out] Низ бајтова ( <b>UChar</b> ) у који се смештају прочитани бајтови.
sizeToEnd	[in/out] Величина којом се одређује колико се бајтова чита (и чија вредност се умањује за број прочитаних бајтова).

## Враћа

1 ако је све добро протекло.  
0 ако је дошло до грешке при читању датотеке.

```
FILE *in = fopen("Ulaz.in", "r");
UChar state[STATE_SIZE];
UInt32 sizeToEnd = 10;
if (!ReadData(in, state, &sizeToEnd))
    printf("Greska pri citanju\n");
```

Дефиниција у линији 524 датотеке aes.c.

2.2.4.16 void RotWord ( UInt32 \* wordPt )

1-бајтна кружна ротација 4-бајтне(**UInt32**) речи улево.

## Параметри

wordPt	[in/out] Показивач на <b>UInt32</b> реч која се ротира.  <b>UInt32</b> t= 0x01; RotWor(&t); // t = 0x0100;
--------	---

Дефиниција у линији 323 датотеке aes.c.

2.2.4.17 void ShiftRows ( UChar state[4] )

Ротира бајтове у врстама, у i-тој врсти за i места улево.

Матрица бајтова state 4 \* 4 трансформише се у нову тако да се свака i-та врста ротира улево i места.

## Параметри

state	[in/out] Матрица бајтова у којој се бајтови по врстама ротирају.  UChar state[4][4] = {{0}, {1}, {2}, {3}} ShiftRows(state); // state[4][4] = {{0}, {0, 0, 0, 1}, {0, 0, 2, 0}, {0, 3, 0, 0}}
-------	--

Дефиниција у линији 98 датотеке aes.c.

2.2.4.18 UInt32 SizeOfFile ( const UChar \* name )

Враћа величину датотеке имена name која се налази у истом фолдеру у којем је програм.

## Параметри

name	[in] Име датотеке чија се величина тражи.
------	---

## Враћа

Величина датотеке у бајтовима.

```
UChar *name = "Ulaz.in";
UInt32 size = SizeOfFile(name); // vraca velicinu datoteke sa imenom name
```

## Упозорење

Величина датотеке не сме бити већа од 4GB. Ако дође до грешке при читању датотеке, непознат је резултат.

Дефиниција у линији 506 датотеке aes.c.

2.2.4.19 `UInt32 StateToUInt32 ( const UChar state[] )`

Преводи низ од 16 бајтова у број од 4 бајта ([UInt32](#)).

Бајтови са нижим индексима у низу иду у више бајтове броја (они који имају већу тежину у броју). Најнижи бајт броја је у `state[STATE_SIZE-1]`. Највиши бајт броја је у `state[STATE_SIZE-4]`.

## Параметри

<code>state</code>	[in/out] Низ бајтова у који се пребацује број.
--------------------	--

## Враћа

4-бајтни број који се добија из `state`.

```
UInt32 t;
UChar state[STATE_SIZE] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1};
t = StateToUInt32(state) // t = 1
```

## Напомена

Функција је неопходна јер се не зна да ли рачунар на ком се покреће програм има LITTLE-↵ENDIAN или BIG-ENDIAN.

Дефиниција у линији 557 датотеке aes.c.

2.2.4.20 `void SubBytes ( UChar state[][4] )`

Мења вредност сваког појединачног бајта из `state` одговарајућом вредношћу из [S\\_BOX](#).

## Параметри

<code>state</code>	[in/out] Матрица 4 *4 бајтова над чијим вредностима се врши замена.  <code>UChar state[4][4] = {0};</code> <code>SubBytes(state);</code> Заменењује сваки бајт из <code>state</code> са одговарајућом вредношћу из <a href="#">S_BOX</a>
--------------------	---

Дефиниција у линији 60 датотеке aes.c.

2.2.4.21 `UInt32 SubWord ( const UInt32 word )`

Замена сваког бајта 4-бајтне речи `word` одговарајућим бајтом из [S\\_BOX](#).

## Параметри

<code>word</code>	[in] 4-бајтна реч типа <a href="#">UInt32</a> код које се мењају бајтови.
-------------------	---

## Враћа

4-бајтна реч која се добија заменом бајтова из `word`

```
UInt32 t = 0x1;
t = SubWord(t); // t = 0x6363637c
```

Дефиниција у линији 327 датотеке aes.c.

2.2.4.22 `UInt32 UCharToUInt32 ( const UChar byte[] )`

Претвара низ 4 бајта у [UInt32](#).

## Параметри

byte	[in] Низ 4 бајта (тип <a href="#">UChar</a> ).
------	--

## Враћа

Враћа број типа [UInt32](#) чија је представа byte

```
UChar byte[4] = { \x0, \x1, \x2, \x3 }
UInt32 t = UCharToUInt32(byte); // t добија вредност 0x0123
```

Дефиниција у линији 8 датотеке aes.c.

2.2.4.23 void UInt32ToState ( const UInt32 num, UChar state[] )

Преводи 4-бајтни број типа [UInt32](#) у низ од 16 бајтова.

Виши бајтови броја (они који имају већу тежину у броју) иду на ниже индексе у низу. Остали бајтови се попуњавају са нулама. Низ мора имати меморију од бар [STATE\\_SIZE](#) бајтова. Најнижи бајт броја се ставља у state[[STATE\\_SIZE](#)-1]

## Параметри

num	[in] Број који пребацујемо у низ бајтова.
state	[in/out] Низ бајтова у који се пребацује број.  <a href="#">UInt32</a> t = 1; <a href="#">UChar</a> state[ <a href="#">STATE_SIZE</a> ]; <a href="#">UInt32ToState</a> (t, state); // poslednji bajt state je 1, svi ostali su 0

## Напомена

Функција је неопходна јер се не зна да ли рачунар на ком се покреће програм има [LITTLE-↵](#) ENDIAN или BIG-ENDIAN.

Дефиниција у линији 549 датотеке aes.c.

2.2.4.24 int WriteData ( FILE \* f, const UChar state[] )

Пише у датотеку f блок бајтова [STATE\\_SIZE](#) из state.

## Параметри

f	[in/out] Показивач на датотеку у који се бајтови пишу.
state	[in] Низ бајтова ( <a href="#">UChar</a> ) који се уписују.

## Враћа

1 ако је све добро протекло.

0 ако је дошло до грешке при писању датотеке.

```
FILE *out = fopen("Izlaz.out", "w");
UChar state[STATE\_SIZE] = {0};
if (!WriteData(out, state)) // pise u datoteku Izlaz.out STATE_SIZE bajtova ciji su svi bitovi 0
printf("Greska pri pisanju\n");
```

Дефиниција у линији 542 датотеке aes.c.

## 2.2.5 Документација променљиве

## 2.2.5.1 const UInt32 CRC32\_TABLE[]

Прерачунате вредности које се користе код провере исправности шифрованог датотеке датим кључем.

Дефиниција у линији 195 датотеке aes.h.

## 2.2.5.2 const UChar G\_FIELD\_MUL[256][6]

Прерачунате вредности множења са 2, 3, A, B, C, D у Галоаовом пољу.

Дефиниција у линији 58 датотеке aes.h.

## 2.2.5.3 const UChar INVS\_BOX[16][16]

Почетна вредност:

```
= {
    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
    0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
    0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
    0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
    0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
    0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
    0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
    0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
    0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
    0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
    0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
}
```

Прерачунате вредности које користи функција InvSubByte

Дефиниција у линији 38 датотеке aes.h.

## 2.2.5.4 const UInt32 ROUND\_CONSTANT[]

Почетна вредност:

```
= { 0x01000000, 0x02000000, 0x04000000, 0x08000000, 0x10000000, 0x20000000,
    0x40000000, 0x80000000, 0x1b000000, 0x36000000, 0x6c000000, 0xd8000000,
    0xab000000, 0x4d000000, 0x9a000000 }
```

Константа која осигурава сигурност експанзије кључа

Дефиниција у линији 190 датотеке aes.h.

## 2.2.5.5 const UChar S\_BOX[16][16]

Почетна вредност:

```
= {
    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
    0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
    0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
    0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
    0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
    0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
    0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
    0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
    0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
    0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
}
```



```
0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,  
0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,  
0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,  
0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,  
0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16  
}
```

Прерачунате вредности које користи функција SubByte.

Дефиниција у линији 18 датотеке aes.h.