

Attack at Dawn

Generated by Doxygen 1.8.14

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	blok Struct Reference	5
3.2	column Struct Reference	5
3.2.1	Detailed Description	6
3.3	couple Struct Reference	6
3.3.1	Detailed Description	6
3.4	field Struct Reference	6
3.4.1	Detailed Description	7
3.5	key Struct Reference	7
3.5.1	Detailed Description	7
3.6	menu Struct Reference	7
3.6.1	Detailed Description	8
3.7	prog Struct Reference	8
3.7.1	Detailed Description	8

4 File Documentation	9
4.1 C:/Users/edvin/Desktop/Attack at Dawn/aadcmd.h File Reference	9
4.1.1 Detailed Description	10
4.1.2 Function Documentation	10
4.1.2.1 aes_cmd()	10
4.1.2.2 batch_mode()	10
4.1.2.3 des_cmd()	11
4.1.2.4 do_batch()	11
4.1.2.5 interpret()	11
4.1.2.6 open_log()	12
4.1.2.7 print_to_log()	12
4.1.2.8 start_cmd_mode()	12
4.1.2.9 triple_des_cmd()	13
4.2 C:/Users/edvin/Desktop/Attack at Dawn/aes.h File Reference	13
4.2.1 Detailed Description	14
4.2.2 Function Documentation	14
4.2.2.1 AddKey()	14
4.2.2.2 Aes_Cipher_Block()	15
4.2.2.3 Aes_Cipher_File()	15
4.2.2.4 Aes_Decipher_Block()	16
4.2.2.5 Aes_Decipher_File()	16
4.2.2.6 ime_provera()	17
4.2.2.7 InverseMixColumns()	17
4.2.2.8 InverseShiftRows()	18
4.2.2.9 InverseSubBytes()	18
4.2.2.10 KeyExpansion()	18
4.2.2.11 MixColumns()	19
4.2.2.12 ShiftRows()	19
4.2.2.13 SubBytes()	19
4.2.2.14 SubWord()	20

4.3	C:/Users/edvin/Desktop/Attack at Dawn/DES.h File Reference	20
4.3.1	Detailed Description	20
4.3.2	Function Documentation	21
4.3.2.1	DES_decrypt_file()	21
4.3.2.2	DES_encrypt_file()	21
4.3.2.3	triple_DES_decrypt_file()	21
4.3.2.4	triple_DES_encrypt_file()	23
4.4	C:/Users/edvin/Desktop/Attack at Dawn/hash.h File Reference	23
4.4.1	Detailed Description	24
4.4.2	Function Documentation	24
4.4.2.1	Dodaj_ime_i_velicinu()	24
4.4.2.2	isGood()	24
4.4.2.3	mojHash()	25
4.4.2.4	procitajHash()	25
4.4.2.5	procitajINFO()	25
4.4.2.6	stringHash()	26
4.4.2.7	upisiHash()	26
4.5	C:/Users/edvin/Desktop/Attack at Dawn/keydecode.h File Reference	26
4.5.1	Detailed Description	27
4.5.2	Function Documentation	27
4.5.2.1	hex2key()	27
4.6	C:/Users/edvin/Desktop/Attack at Dawn/main.c File Reference	27
4.6.1	Detailed Description	30
4.6.2	Function Documentation	30
4.6.2.1	alg2string()	30
4.6.2.2	ed_dest_func()	31
4.7	C:/Users/edvin/Desktop/Attack at Dawn/menus.h File Reference	31
4.7.1	Detailed Description	34
4.7.2	Function Documentation	34
4.7.2.1	add_column()	34

4.7.2.2	<code>add_field()</code>	35
4.7.2.3	<code>compact()</code>	35
4.7.2.4	<code>cursor_down()</code>	35
4.7.2.5	<code>cursor_left()</code>	36
4.7.2.6	<code>cursor_right()</code>	36
4.7.2.7	<code>cursor_up()</code>	36
4.7.2.8	<code>end_prog()</code>	37
4.7.2.9	<code>free_field()</code>	37
4.7.2.10	<code>get_active_menu()</code>	37
4.7.2.11	<code>get_column()</code>	37
4.7.2.12	<code>get_f()</code>	38
4.7.2.13	<code>get_field()</code>	38
4.7.2.14	<code>get_menu()</code>	39
4.7.2.15	<code>getPos()</code>	39
4.7.2.16	<code>init_prog()</code>	39
4.7.2.17	<code>input_box()</code>	40
4.7.2.18	<code>lockColumn()</code>	40
4.7.2.19	<code>message_box()</code>	40
4.7.2.20	<code>new_column()</code>	41
4.7.2.21	<code>new_field()</code>	41
4.7.2.22	<code>new_menu()</code>	42
4.7.2.23	<code>next_menu()</code>	42
4.7.2.24	<code>nothing()</code>	42
4.7.2.25	<code>prev_menu()</code>	43
4.7.2.26	<code>print_column()</code>	43
4.7.2.27	<code>print_field()</code>	43
4.7.2.28	<code>print_menu()</code>	44
4.7.2.29	<code>print_title()</code>	44
4.7.2.30	<code>remove_field()</code>	44
4.7.2.31	<code>remove_selected_field()</code>	45
4.7.2.32	<code>run_prog()</code>	45
4.7.2.33	<code>set_border()</code>	45
4.7.2.34	<code>set_field_string()</code>	46
4.7.2.35	<code>set_update()</code>	46
4.7.2.36	<code>update_menu()</code>	46

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

blok	5
column	Struktura koja definise sve potrebne informacije za kolonu Kolona se sastoji od niza vertikalno raspoređenih polja	5
couple	Pomocna struktura za povezivanje kolona files keys i algorithm u encryption decryption podmeniju	6
field	Struktura koja definise sve potrebne informacije za polje	6
key	Struktura koja služi za skladištenje ključa u hexadecimalnoj formi zajedno sa imenom	7
menu	Struktura koja definise sve potrebne informacije za Meni Meni se sastoji iz horizontalno postavljenih susednih kolona	7
prog	Struktura koja definise sve potrebne informacije za sam program koji se pokrece	8

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

C:/Users/edvin/Desktop/Attack at Dawn/ aadcmd.h	
Header sa svim funkcijama za konzolni mod Header fajl sa svim funkcijama koji su potrebni za pokretanje instrukcija pozvanih pomocu konzole	9
C:/Users/edvin/Desktop/Attack at Dawn/ aes.h	
Funkcije koriscenje u enkripciji i dekripciji koriscenja Aes standarda	13
C:/Users/edvin/Desktop/Attack at Dawn/ DES.h	
Funkcije za enkripciju i dekripciju fajlova DES ili triple-DES algoritmom	20
C:/Users/edvin/Desktop/Attack at Dawn/ hash.h	
Funkcije za rad sa hashom koji služi za proveru validnosti ključa, kao i funkcije za upis traženih informacija u fajl	23
C:/Users/edvin/Desktop/Attack at Dawn/ keydecode.h	
Sve funkcije za konvertovanje ključa Header file koji sadrži sve funkcije vezane za konverzije ključa iz jednog oblika u drugi	26
C:/Users/edvin/Desktop/Attack at Dawn/ main.c	
Funkcije vezane za učitavanje GUI elemenata i njihovo korišćenje zajedno sa main funkcijom	27
C:/Users/edvin/Desktop/Attack at Dawn/ menus.h	
Sve funkcije vezane za iscrtavanje i održavanje menija Header fajl koji sadrži sve funkcije koje programer može da koristi za kreiranje menija i raspoređivanje GUI elemenata kao i za unos od strane korisnika Oslanja se na biblioteku PDCURSES	31

Chapter 3

Data Structure Documentation

3.1 blok Struct Reference

Data Fields

- unsigned char **a** [8]

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/DES.c

3.2 column Struct Reference

Struktura koja definiše sve potrebne informacije za kolonu Kolona se sastoji od niza vertikalno raspoređenih polja.

```
#include <menus.h>
```

Data Fields

- int **number_of_fields**
- int **number_of_selected**
- int **width**
- int **hasBorder**
- int **a_bor**
- int **b_bor**
- int **is_locked**
- int **lock_pos**
- int **isCompact**
- int **is_designated**
- WINDOW * **win**
- char **column_name** [MAX_NAME_LEN]
- char **column_title** [MAX_NAME_LEN]
- **FIELD** * **first**
- **FIELD** * **last**
- **FIELD** * **first_visible**
- **FIELD** * **first_selected**
- **FIELD** * **last_selected**

3.2.1 Detailed Description

Struktura koja definise sve potrebne informacije za kolonu Kolona se sastoji od niza vertikalno rasporedjenih polja.

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/[menus.h](#)

3.3 couple Struct Reference

pomocna struktura za povezivanje kolona files keys i algorithm u encryption decryption podmeniju

Data Fields

- [FIELD](#) * **file_field**
- [FIELD](#) * **key_field**
- [FIELD](#) * **alg_field**
- char **file_name** [MAX_INPUT_LEN+1]
- [KEY](#) * **key**
- [Algorithm](#) **alg**

3.3.1 Detailed Description

pomocna struktura za povezivanje kolona files keys i algorithm u encryption decryption podmeniju

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/[main.c](#)

3.4 field Struct Reference

Struktura koja definise sve potrebne informacije za polje.

```
#include <menus.h>
```

Data Fields

- void * **x**
- void(* **f**)(void)
- void(*)(*) **free_func** (void *)
- char **field_name** [MAX_FIELD_NAME_LEN]
- char **field_string** [MAX_FIELD_STRING_LEN]
- int **is_selected**
- int **position**
- struct [field](#) * **next**
- struct [field](#) * **prev**
- struct [field](#) * **next_select**
- struct [field](#) * **prev_select**

3.4.1 Detailed Description

Struktura koja definise sve potrebne informacije za polje.

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/[menus.h](#)

3.5 key Struct Reference

struktura koja služi za skladištenje ključa u hexadecimalnoj formi zajedno sa imenom

Data Fields

- char **name** [MAX_KEY_NAME_LEN]
- char **key** [MAX_KEY_LEN]
- int **is_named**

3.5.1 Detailed Description

struktura koja služi za skladištenje ključa u hexadecimalnoj formi zajedno sa imenom

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/[main.c](#)

3.6 menu Struct Reference

Struktura koja definise sve potrebne informacije za Meni. Meni se sastoji iz horizontalno postavljenih susednih kolona.

```
#include <menus.h>
```

Data Fields

- int **number_of_columns**
- int **start_col**
- [COLUMN](#) * **columns** [MAX_COLUMNS]
- char * **column_names** [MAX_COLUMNS]
- char **menu_name** [MAX_NAME_LEN]
- [FIELD](#) * **cursor_field**
- [FIELD](#) * **special_field**
- [COLUMN](#) * **cursor_column**

3.6.1 Detailed Description

Struktura koja definise sve potrebne informacije za Meni Meni se sastoji iz horizontalno postavljenih sustednih kolona.

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/[menus.h](#)

3.7 prog Struct Reference

Struktura koja definise sve potrebne informacije za sam program koji se pokrece.

```
#include <menus.h>
```

Data Fields

- void(* **update**)(void)
- int **number_of_menus**
- int **top**
- int **is_stopped**
- int **column_height**
- int **max_width**
- [MENU](#) * **menus** [MAX_MENUS]
- char * **menu_names** [MAX_MENUS]
- [MENU](#) * **menu_stack** [MAX_MENUS]

3.7.1 Detailed Description

Struktura koja definise sve potrebne informacije za sam program koji se pokrece.

The documentation for this struct was generated from the following file:

- C:/Users/edvin/Desktop/Attack at Dawn/[menus.h](#)

Chapter 4

File Documentation

4.1 C:/Users/edvin/Desktop/Attack at Dawn/aadcmd.h File Reference

Header sa svim funkcijama za konzolni mod Header fajl sa svim funkcijama koji su potrebni za pokretanje instrukcija pozvanih pomocu konzole.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <string.h>
#include <ctype.h>
#include "aes.h"
#include "DES.h"
#include "keydecode.h"
#include <time.h>
```

Functions

- void [start_cmd_mode](#) (int argc, char **argv)
Zapocinje konzolni mod rada.
- void [interpret](#) (int argc, char **argv, int is_batch, FILE **log_file)
Interpretira instrukciju.
- FILE * [open_log](#) ()
Zapocinje log fajl.
- void [print_to_log](#) (char *outstr, int is_batch, FILE **log_file)
Ispisuje zadatu poruku.
- void [batch_mode](#) (int argc, char **argv, FILE **log_file)
Zapocinje batch rezim rada.
- void [do_batch](#) (char *batch_file, FILE **log_file)
Izvršava instrukcije u batch rezimu.
- void [triple_des_cmd](#) (int argc, char **argv, int is_batch, FILE **log_file)
parsira komandu 3des i pokrece triple des
- void [des_cmd](#) (int argc, char **argv, int is_batch, FILE **log_file)
parsira komandu 3des i pokrece des
- void [aes_cmd](#) (int argc, char **argv, int is_batch, FILE **log_file)
parsira komandu 3des i pokrece triple aes

4.1.1 Detailed Description

Header sa svim funkcijama za konzolni mod Header fajl sa svim funkcijama koji su potrebni za pokretanje instrukcija pozvanih pomocu konzole.

Author

Edvin Maid

4.1.2 Function Documentation

4.1.2.1 aes_cmd()

```
void aes_cmd (
    int argc,
    char ** argv,
    int is_batch,
    FILE ** log_file )
```

parsira komandu 3des i pokrece triple aes

Parameters

<i>argc</i>	broj argumenata za komandu aes
<i>argv</i>	argumenti za komandu aes
<i>is_batch</i>	1 ako je batch rezim rada ukljucen 0 ako je iskljucen
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi Parsira komandu i njene argumente, proverava ispravnost, ukoliko dodje do greske ispise je koriscenjem funkcije print_to_log() Uspesno parsirane komande pokrece

4.1.2.2 batch_mode()

```
void batch_mode (
    int argc,
    char ** argv,
    FILE ** log_file )
```

Zapocinje batch rezim rada.

Parameters

<i>argc</i>	broj fajlova koje treba izvorsiti u batch rezimu
<i>argv</i>	destinacije fajlova koje treba izvorsiti u batch rezimu
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi Funkcija zapocinje batch rezim rada i za svaki fajl pokrece do_batch

4.1.2.3 des_cmd()

```
void des_cmd (
    int argc,
    char ** argv,
    int is_batch,
    FILE ** log_file )
```

parsira komandu 3des i pokrece des

Parameters

<i>argc</i>	broj argumenata za komandu des
<i>argv</i>	argumenti za komandu des
<i>is_batch</i>	1 ako je batch rezim rada ukljucen 0 ako je iskljucen
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi Parsira komandu i njene argumente, proverava ispravnost, ukoliko dodje do greske ispise je koriscenjem funkcije print_to_log() Uspesno parsirane komande pokrece

4.1.2.4 do_batch()

```
void do_batch (
    char * batch_file,
    FILE ** log_file )
```

Izvrsva instrukcije u batch rezimu.

Parameters

<i>batch_file</i>	putanja do fajla koji izvrsva
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi Parsira zadati fajl i deli ga u niz instrukcija koje prosledjuje interpret funkciji

4.1.2.5 interpret()

```
void interpret (
    int argc,
    char ** argv,
    int is_batch,
    FILE ** log_file )
```

Interpretira instrukciju.

Parameters

<i>argc</i>	broj argumenata
<i>argv</i>	niz argumenata bez imena programa
<i>is_batch</i>	1 ako je batch rezim rada ukljucen 0 ako je iskljucen
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi

4.1.2.6 open_log()

```
FILE* open_log ( )
```

Zapocinje log fajl.

Returns

Pokazivac na log fajl Otvara novi fajl u rezimu "w" u kojem ce se upisivati sve poruke za greske koje budu nastale u batch rezimu rada

4.1.2.7 print_to_log()

```
void print_to_log (
    char * outstr,
    int is_batch,
    FILE ** log_file )
```

Ispisuje zadatu poruku.

Parameters

<i>outstr</i>	Poruka koju ispisuje
<i>is_batch</i>	1 ako je batch rezim rada ukljucen 0 ako je iskljucen
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi Ispisuje zadatu poruku na standardni izlaz ukoliko je <i>is_batch</i> = 0 Ispisuje zadatu poruku u log fajl ukoliko je <i>is_batch</i> = 1 Ukoliko fajl nije kreiran poziva <i>open_log</i>

4.1.2.8 start_cmd_mode()

```
void start_cmd_mode (
    int argc,
    char ** argv )
```

Zapocinje konzolni mod rada.

Parameters

<i>argc</i>	broj argumenata
<i>argv</i>	niz argumenata koje program obradjuje bez imena programa Zapocinje konzolni mod rada i prosledjuje zadate instrukcije i parametre i izvrsava ih

4.1.2.9 triple_des_cmd()

```
void triple_des_cmd (
    int argc,
    char ** argv,
    int is_batch,
    FILE ** log_file )
```

parsira komandu 3des i pokrece triple des

Parameters

<i>argc</i>	broj argumenata za komandu 3des
<i>argv</i>	argumenti za komandu 3des
<i>is_batch</i>	1 ako je batch rezim rada ukljucen 0 ako je iskljucen
<i>log_file</i>	fajl u koji ce se upisati poruke ukoliko dodje do greske u obradi Parsira komandu i njene argumente, proverava ispravnost, ukoliko dodje do greske ispise je koriscenjem funkcije print_to_log() Uspesno parsirane komande pokrece

4.2 C:/Users/edvin/Desktop/Attack at Dawn/aes.h File Reference

Funkcije koriscenje u enkripciji i dekripciji koriscenja Aes standarda.

Functions

- unsigned int * [KeyExpansion](#) (int Nk, unsigned char *key, int Nr)
Funkcija za kreiranje round blokova koji se koriste u Aes enkripciji i dekripciji.
- int [SubBytes](#) (unsigned char **state)
*Funkcija koja za svaki byte u **state** matrici zameni sa onom na koju pokazuje u tabeli S-box.*
- unsigned int [SubWord](#) (unsigned int a)
Funkcija koja za svaki byte u reci od 4 bajta zameni sa onom koji se nalazu u tabeli S-box.
- int [ShiftRows](#) (unsigned char **state)
*Funkcija koja vrsi pomeranje redova u **state** matrici 4x4.*
- int [MixColumns](#) (unsigned char **state)
*Funkcija koja vrsi razne promene nad kolonama **state** matrice.*
- int [InverseShiftRows](#) (unsigned char **state)
*Funkcija koja vrsi inverzno pomeranje od funkcije **ShiftRows**.*
- int [InverseMixColumns](#) (unsigned char **state)
*Funkcija koja radi inverzno od funkcije **MixColumns**.*

- int [InverseSubBytes](#) (unsigned char **state)
*Funkcija koja radi inverzno od funkcije **SubBytes**.*
- int [AddKey](#) (unsigned char **state, unsigned int *word)
*Xor-uje round blok(niz od 4 reci duzine 4-bajta) i **state** matricu.*
- char * [ime_provera](#) (char *ime_fajla)
Funkcija proverava da li lokacija fajla koja je poslata postoji i ako postoji onda ga menja.
- int [Aes_Cipher_Block](#) (unsigned int *word, unsigned char *input, unsigned char *output, int Nr)
Funkcija koji enkriptuje blok memorije duzine 128-bita.
- int [Aes_Decipher_Block](#) (unsigned int *word, unsigned char *input, unsigned char *output, int Nr)
Funkcija koji dekriptuje blok memorije duzine 128-bita.
- int [Aes_Decipher_File](#) (char *ime_ciphera, char *key_name, int Nk, char *path)
Funkcija koja vrsi dekripciju fajla.
- int [Aes_Cipher_File](#) (char *ime_fajla, char *key_name, int Nk, char *path)
Funkcija koja vrsi enkripciju fajla.

4.2.1 Detailed Description

Funkcije koriscenje u enkripciji i dekripciji koriscenja Aes standarda.

Ovaj header fajl sadrzi funkcije koriscenje prilikom enkripciji i dekripciji Aes fajlova kljucem duzina 128,192,256 bita duzine.

Author

Jovan Spasojevic

4.2.2 Function Documentation

4.2.2.1 AddKey()

```
int AddKey (
    unsigned char ** state,
    unsigned int * word )
```

Xor-uje round blok(niz od 4 reci duzine 4-bajta) i **state** matricu.

Parameters

in	<i>word</i>	Niz od 4 reci duzine 4-bajta koji pretstavlja jedan round blok
in, out	<i>state</i>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno dekriptuje

Funkcija XOR-uje svaku kolonu iz **state** matrice 4x4 sa reci i niza **word** (Column[i]=Column[i] Xor word[i]). Ova funkcija je sama sebi inverzna.

4.2.2.2 Aes_Cipher_Block()

```
int Aes_Cipher_Block (
    unsigned int * word,
    unsigned char * input,
    unsigned char * output,
    int Nr )
```

Funkcija koji enkriptuje blok memorije duzine 128-bit.

Parameters

in	<i>input</i>	Blok memorije koji treba enkriptovati
in, out	<i>output</i>	Blok memorije koji je enkriptovan
in	<i>Nr</i>	Broj rundi koje rade Aes enkripcija i dekripcija
in	<i>word</i>	Niz reci od (Nr+1)*4 duzine

Funkcija koristi prethodne definisane funkcije **ShiftRows**, **MixColumns**, **AddKey**, **SubBytes** da vrsi razne transformacije **state** matrice kreirane deljenjem **input** niza po kolonama(`state[j][i]=word[i*4+j]`). Na kraju se **state** matrica razdeli na kolone i sve spoje u niz **output** (`output[i*4+j]=state[j][i]`). Funkcija vraca pokazivac na taj niz.

See also

[ShiftRows](#)
[MixColumns](#)
[AddKey](#)
[SubBytes](#)

4.2.2.3 Aes_Cipher_File()

```
int Aes_Cipher_File (
    char * ime_fajla,
    char * key_name,
    int Nk,
    char * path )
```

Funkcija koja vrsi enkripciju fajla.

Parameters

in	<i>ime_fajla</i>	Lokacija fajla koji treba da bude enkriptovan
in	<i>key_name</i>	Kljuc koji se koristi u enkripciji
in	<i>Nk</i>	Broj koji predstavlja duzinu bloka u bajtovima podeljeni sa 4
in	<i>path</i>	Nova lokacija enkriptovanog fajla dobijenog ovom funkcijom(moze biti NULL)

Fajl na lokaciji **ime_fajla** pre citanja se trazi njegovo ime i velicina. Posto se nadju kreira se novi fajl koji od oblika imena file.extension pravi novi fajl file_aes.txt. Proverava se ime kreiranog fajla u slucaju da vec postoji na lokaciji gde treba da se smesti i ako postoji menja svoje ime. Potom se cita fajl na lokaciji <ime_fajla> u blokovima od po 8KB i onda se dele na delove od po 16 bajta koji se enkriptuju funkcijom **Aes_Cipher_Block** i upisuju u file_aes.txt.

Na kraju programa se pomocu funkcija iz [hash.h](#) dodaju 8 bajta na kraju fajla koji se koriste za identifikaciju ključa. Funkcija ako je path NULL enkriptovan fajl stavlja gde je i sam program, a ako path postoji onda ga postavlja na tu lokaciju. Funkcija takodje na pocetak fajla stavlja njegovo originalno ime bez lokacije i njegovu duzinu.

4.2.2.4 Aes_Decipher_Block()

```
int Aes_Decipher_Block (
    unsigned int * word,
    unsigned char * input,
    unsigned char * output,
    int Nr )
```

Funkcija koji dekriptuje blok memorije duzine 128-bita.

Parameters

in	<i>input</i>	Blok memorije koji je enkriptovan
in, out	<i>output</i>	Blok memorije koji je dekriptovan
in	<i>Nr</i>	Broj rundi koje rade Aes enkripcija i dekripcija
in	<i>word</i>	Niz reci od (Nr+1)*4 duzine

Funkcija koristi prethodne definisane funkcije **InverseShiftRows**, **InverseMixColumns**, **InverseAddKey**, **InverseSubBytes** da vrsi razne transformacije **state** matrice kreirane deljenjem **input** niza po kolonama ($state[j][i] = word[i*4+j]$). Na kraju se **state** matrica razdeli na kolone i sve spoje u niz **output** ($output[i*4+j] = state[j][i]$). Funkcija vraca pokazivac na taj niz.

See also

[InverseShiftRows](#)
[InverseMixColumns](#)
[AddKey](#)
[InverseSubBytes](#)

4.2.2.5 Aes_Decipher_File()

```
int Aes_Decipher_File (
    char * ime_ciphera,
    char * key_name,
    int Nk,
    char * path )
```

Funkcija koja vrsi dekripciju fajla.

Parameters

in	<i>ime_ciphera</i>	Lokacija fajla koji treba da bude dekriptovan
in	<i>key_name</i>	Kljuc koji se koristi u dekripciji
in	<i>Nk</i>	Broj koji predstavlja duzinu reci u bajtovima podeljeni sa 4
in	<i>path</i>	Nova lokacija dekriptovanog fajla dobijenog ovom funkcijom(moze biti NULL)

Pre pocetka citanje u blokovima, funkcija na pocetku procita 8 bajta na kraju fajla **ime_ciphera** koji predstavljaju Mac koji se koristi za proveru da li je kljuc odgovarajuci. Ako jeste nastavi se dekripcija ako nije vrati izlaz kao 1. Takodje cita pocetak fajla gde se nalaze ime fajla u kome treba da se upisu dekriptovani podaci i kao njegova velicina. Posle toga cita fajl na lokaciji **ime_ciphera** u blokovima od po 8kb i onda deli taj blok na delove od po 16 bajta i koristi **Aes_Decipher_Block** da dekriptira te blokove. Funkcija ako je path NULL dekriptovan fajl stavlja gde je i sam program, a ako path postoji onda ga postavlja na tu lokaciju. Fajl nece vrsiti dekripciju ako je unet pogresan kljuc. Ime fajla i njegova lokacija se proveravaju pre dekripcije da se utvrdi da li postoji fajl sa istim imenom i menja svoje ime ako postoji.

See also

[Aes_Decipher_Block](#)
[ime_provera](#)

4.2.2.6 ime_provera()

```
char* ime_provera (
    char * ime_fajla )
```

Funkcija proverava da li lokacija fajla koja je poslata postoji i ako postoji onda ga menja.

Parameters

in	<i>ime_fajla</i>	Lokacija fajla koji se proverava
----	------------------	----------------------------------

Funkcija vraća Lokaciju fajla sa promenjenim imenom(ako fajl sa takvim imenom vec postoji),ili sa ne promenjenim.

4.2.2.7 InverseMixColumns()

```
int InverseMixColumns (
    unsigned char ** state )
```

Funkcija koja radi inverzno od funkcije **MixColumns**.

Parameters

in, out	<i>state</i>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno dekriptuje
---------	--------------	--

Radi slicno kao **MixColumns** samo sto se koriste druga konstantna matrica 4x4 i 4x1.

See also

[MixColumns](#)

4.2.2.8 InverseShiftRows()

```
int InverseShiftRows (
    unsigned char ** state )
```

Funkcija koja vrsi inverzno pomeranje od funkcije **ShiftRows**.

Parameters

<i>in, out</i>	<i>state</i>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno dekriptuje
----------------	--------------	--

Funkcija radi slicno kao **ShiftRows** samo sto se pomeranja desavaju udesno.

See also

[ShiftRows](#)

4.2.2.9 InverseSubBytes()

```
int InverseSubBytes (
    unsigned char ** state )
```

Funkcija koja radi inverzno od funkcije **SubBytes**.

Parameters

<i>in, out</i>	<i>state</i>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno dekriptuje
----------------	--------------	--

Radi slicno kao **SubBytes** samo sto umesto S-box tabele koristi Inverse S-box tabelu.

See also

[SubBytes](#)

4.2.2.10 KeyExpansion()

```
unsigned int* KeyExpansion (
    int Nk,
    unsigned char * key,
    int Nr )
```

Funkcija za kreiranje round blokova koji se koriste u Aes enkripciji i dekripciji.

Parameters

in	<i>Nk</i>	Predstavlja broj pocetnih reci duzine 4-bajta
in	<i>key</i>	Kljuc od koga se prave svi round blokovi duzine 4 reci
in	<i>Nr</i>	Broj rundi koji se desavaju u Aes enkripciji i dekripciji
out	<i>word</i>	Niz reci svi duzine 4-bajta

Funkcija na pocetku ucita **Nk** broj reci duzine 4-bajta iz **key** niza bajtova u niz **word**. Onda na osnovu 4 poslednje reci iz niza **word** putem raznih transformacija napravi 4 nove reci koje dodaje u niz. To radi dok niz nema $(Nr+1)*4$ reci u nizu. Na zavrsetku funkcije se vraca pokazivac na taj niz.

4.2.2.11 MixColumns()

```
int MixColumns (
    unsigned char ** state )
```

Funkcija koja vrsi razne promene nad kolonama **state** matrice.

Parameters

in, out	<i>state</i>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno enkriptuje
---------	--------------	--

Matrica **state** se podeli na 4 kolone. Svaka kolona ce se mnoziti sa 4x4 konstantnom matricom. Novim proizvedenim matricicama 4x1 tj. kolonama se dodaje konstantna kolona. Te nove kolone se posle spajaju i zamenjuju **state** matricu.

4.2.2.12 ShiftRows()

```
int ShiftRows (
    unsigned char ** state )
```

Funkcija koja vrsi pomeranje redova u **state** matrici 4x4.

Parameters

in, out	<i>state</i>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno enkriptuje
---------	--------------	--

Funkcija vrsi pomeranje redova tako sto pomeri tako sto za 2. red matrice sve njene clanove pomeri ulevo,tako da $state[i][0]=state[i][1]$, $state[i][1]=state[i][2]$, $state[i][2]=state[i][3]$ i $state[i][3]=state[i][0]$. 3. red se pomera 2 puta ulevo,4. red 3 puta ulevo na isti nacin.

4.2.2.13 SubBytes()

```
int SubBytes (
    unsigned char ** state )
```

Funkcija koja za svaki byte u **state** matrici zameni sa onom na koju pokazuje u tabeli S-box.

Parameters

<code>in, out</code>	<code>state</code>	Matrica 4x4 koja sadrzi trenutno stanje bloka memorije koji se trenutno enkriptuje
----------------------	--------------------	--

Za svaki bajt u matrici **state** trazi se njegova vrednost u tabeli S-box(`S_Box[state[i][j]]`) i onda vrednost u matrici biva zamenjen vrednoscu u S-Box-u.

4.2.2.14 SubWord()

```
unsigned int SubWord (
    unsigned int a )
```

Funkcija koja za svaki byte u reci od 4 bajta zameni sa onom koji se nalazu u tabeli S-box.

Parameters

<code>in</code>	<code>a</code>	Kopija reci duzine 4 bajta kojoj treba menjati vrednosti
<code>out</code>	<code>resenje</code>	Nova kreirana rec koriscenjem a

Funkcija podeli rec **a** duzine 4-bajta na 4 dela. Onda se za svaki taj deo trazi njegova vrednost u S-Box-u i onda sve sve te nove vrednosti spoje u novu rec **resenje**. Funkcija na kraju vraca **resenje** kao izlaz.

4.3 C:/Users/edvin/Desktop/Attack at Dawn/DES.h File Reference

Funkcije za enkripciju i dekripciju fajlova DES ili triple-DES algoritmom.

Functions

- int [DES_encrypt_file](#) (char *path, char *c, char *dest)
Funkcija za enkripciju fajla koristeći DES algoritam.
- int [DES_decrypt_file](#) (char *path, char *c, char *dest)
Funkcija za dekripciju fajla koristeći DES algoritam.
- int [triple_DES_encrypt_file](#) (char *path, char *c, char *dest)
Funkcija za enkripciju fajla koristeći tripe-DES algoritam.
- int [triple_DES_decrypt_file](#) (char *path, char *c, char *dest)
Funkcija za dekripciju fajla koristeći triple-DES algoritam.

4.3.1 Detailed Description

Funkcije za enkripciju i dekripciju fajlova DES ili triple-DES algoritmom.

Author

Andrija Kolic Header fajl koji sadrzi sve funkcije koje korisnik moze da pozove koristeći interfejs ili konzolu. Sadrzi funkcije za enkripciju i dekripciju fajla koristeći DES algoritam, kao i funkcije za enkripciju i dekripciju fajla koristeći triple-DES algoritam.

4.3.2 Function Documentation

4.3.2.1 DES_decrypt_file()

```
int DES_decrypt_file (
    char * path,
    char * c,
    char * dest )
```

Funkcija za dekripciju fajla koristeći DES algoritam.

Parameters

in	<i>path</i>	Ime enkriptovanog fajla
in	<i>c</i>	Kljuc koji se koristi za dekripciju, priložen u vidu niza char-ova dužine 8
in	<i>dest</i>	Destinacija dekriptovanog fajla, rezultujući fajl će biti kreiran na toj lokaciji

Funkcija dekriptuje fajl sa imenom 'path' i ne menja dati string. Koristi ključ koji učitava iz stringa c, ne menjajući njegov sadržaj. Rezultujući fajl će se nalaziti na lokaciji 'dest' i imaće ime originala, koje je upisano u enkriptovani fajl, možda neznatno izmenjeno. Funkcija vraća 0 ako je enkriptovanje uspešno izvršeno, vraća -1 ako otvaranje fajla nije uspešno, a vraća 1 ako priloženi ključ nije odgovarajući.

4.3.2.2 DES_encrypt_file()

```
int DES_encrypt_file (
    char * path,
    char * c,
    char * dest )
```

Funkcija za enkripciju fajla koristeći DES algoritam.

Parameters

in	<i>path</i>	Ime fajla koji se enkriptuje
in	<i>c</i>	Ključ koji se koristi za enkripciju, priložen u vidu niza char-ova dužine 8
in	<i>dest</i>	Destinacija enkriptovanog fajla, rezultujući fajl će biti kreiran na toj lokaciji

Funkcija enkriptuje fajl sa imenom 'path' i ne menja dati string. Koristi ključ koji učitava iz stringa c, ne menjajući njegov sadržaj. Rezultujući fajl će se nalaziti na lokaciji 'dest' i imaće ime originala, možda neznatno izmenjeno. Funkcija vraća 0 ako je enkriptovanje uspešno izvršeno, a vraća -1 ako otvaranje fajla nije uspešno.

4.3.2.3 triple_DES_decrypt_file()

```
int triple_DES_decrypt_file (
    char * path,
    char * c,
    char * dest )
```

Funkcija za dekripciju fajla koristeći triple-DES algoritam.

Parameters

in	<i>path</i>	Ime enkriptovanog fajla
in	<i>c</i>	Kljuc koji se koristi za dekrpciju, prilozen u vidu niza char-ova duzine 16
in	<i>dest</i>	Destinacija dekrptovanog fajla, rezultujuci fajl ce biti kreiran na toj lokaciji

Funkcija dekriptuje fajl sa imenom 'path' i ne menja dati string. Koristi kljuc koji učitava iz stringa c, ne menjajući njegov sadržaj. Rezultujući fajl će se nalaziti na lokaciji 'dest' i imace ime originala, koje je upisano u enkriptovani fajl, mozda neznatno izmenjeno. Funkcija vraca 0 ako je enkriptovanje uspesno izvršeno, vraca -1 ako otvaranje fajla nije uspelo, a vraca 1 ako prilozeni kljuc nije odgovarajuci.

4.3.2.4 triple_DES_encrypt_file()

```
int triple_DES_encrypt_file (
    char * path,
    char * c,
    char * dest )
```

Funkcija za enkripciju fajla koristeći tripe-DES algoritam.

Parameters

in	<i>path</i>	Ime fajla koji se enkriptuje
in	<i>c</i>	Kljuc koji se koristi za enkripciju, prilozen u vidu niza char-ova duzine 16
in	<i>dest</i>	Destinacija enkriptovanog fajla, rezultujuci fajl ce biti kreiran na toj lokaciji

Funkcija enkriptuje fajl sa imenom 'path' i ne menja dati string. Koristi kljuc koji učitava iz stringa c, ne menjajući njegov sadržaj. Rezultujući fajl će se nalaziti na lokaciji 'dest' i imace ime originala, mozda neznatno izmenjeno. Funkcija vraca 0 ako je enkriptovanje uspesno izvršeno, a vraca -1 ako otvaranje fajla nije uspelo.

4.4 C:/Users/edvin/Desktop/Attack at Dawn/hash.h File Reference

Funkcije za rad sa hashom koji služi za proveru validnosti ključa, kao i funkcije za upis traženih informacija u fajl.

Functions

- int [stringHash](#) (char *path, char fajlVeclmaHash)
Funkcija koja racuna hash za enkriptovani text.
- long long [mojHash](#) (char *path, char fajlVeclmaHash, char *kljuc, char duzinaKljuca, int metod)
Funkcija koja racuna hash kod u zavisnosti od fajla, koriscenog kljuca, i metoda enkripcije.
- long long [procitajHash](#) (char *path)
Funkcija koja cita hash kod iz fajla.
- void [upisiHash](#) (char *path, long long hes)
Upisuje hash kod u fajl.
- int [isGood](#) (char *path, char *kljuc, char duzinaKljuca, int metod)
Funkcija proverava da li su dati kljuc i metod odgovarajuci za dati fajl.
- int [Dodaj_ime_i_velicinu](#) (char *ime_fajla, char *ime_dest)
*Funkcija kreira fajl na lokaciji **ime_dest** i dodaje ime i velicinu fajla sa lokacije **ime_fajla***
- void [procitajINFO](#) (char *path, char *ime, int *velicina, int *pocetakFajla)
Funkcija cita ime i velicinu originala iz enkriptovanog fajla.

4.4.1 Detailed Description

Funkcije za rad sa hashom koji služi za proveru validnosti ključa, kao i funkcije za upis traženih informacija u fajl.

Author

Andrija Kolic feat Jovan Spasojevic ovo će opet da se pojavi kod autora, ne? Kada korisnik nije siguran koji ključ/metod je koristio pri enkripciji, dolazi do problema. Da bismo resili ovaj problem svaki enkriptovani fajl ima kod od 8 bajtova na svom kraju. Taj kod je hash funkcija koja prima enkriptovani text, ključ, kao i metod enkripcije. Zbog prirode hash funkcija iz ovog koda se ne može direktno izračunati korišćeni ključ, ali korisnik može proveriti da li koristi ispravan ključ. Pored funkcija vezanih za hash, imamo i dve funkcije koje upisuju/citaju ime i veličinu originalnog fajla u enkriptovani.

4.4.2 Function Documentation

4.4.2.1 Dodaj_ime_i_velicinu()

```
int Dodaj_ime_i_velicinu (
    char * ime_fajla,
    char * ime_dest )
```

Funkcija kreira fajl na lokaciji **ime_dest** i dodaje ime i veličinu fajla sa lokacije **ime_fajla**

Parameters

in	<i>ime_fajla</i>	Lokacija fajla čije se cita ime i velicina
in	<i>ime_dest</i>	Lokacija fajla koji se stvara i gde se velicina i ime proslog fajla upisuju u njega Funkcija prvo kreira fajl na lokaciji ime_dest , a onda upisuje u njega samo ime fajla na lokaciji <ime_fajla> kao i njegovu velicinu.

4.4.2.2 isGood()

```
int isGood (
    char * path,
    char * kljuc,
    char duzinaKljuca,
    int metod )
```

Funkcija proverava da li su dati ključ i metod odgovarajući za dati fajl.

Parameters

in	<i>path</i>	Ime fajla čiji se hash kod proverava
in	<i>kljuc</i>	Vrednost ključa za koji se generise hash kod
in	<i>duzinaKljuca</i>	duzina ključa u bajtovima
in	<i>metod</i>	Pridružena vrednost metoda (pogledati dokumentaciju funkcije mojHash)

Funkcija upoređuje vrednost hash koda upisanog u fajl sa imenom path sa vrednoscu funkcije `mojHash(path,1,kljuc,duzinaKljuca,metod)`. Vraca 1 ako su vrednosti iste, 0 u suprotnom.

4.4.2.3 `mojHash()`

```
long long mojHash (
    char * path,
    char fajlVecImaHash,
    char * kljuc,
    char duzinaKljuca,
    int metod )
```

Funkcija koja racuna hash kod u zavisnosti od fajla, koriscenog kljuca, i metoda enkripcije.

Parameters

in	<i>path</i>	Ime enkriptovanog fajla za koji se racuna hash kod
in	<i>fajlVecImaHash</i>	Ima vrednost 1 ako prilozeni fajl vec ima upisan kod na kraju, vrednost 0 u suprotnom
in	<i>kljuc</i>	Kljuc koriscen pri enkripciji, ili kljuc za koji se proverava ispravnost
in	<i>duzinaKljuca</i>	duzina kljuca (ko bi reko)
in	<i>metod</i>	Metod koriscen za enkripciju, svakom metodu se pridruzuje odredjen broj

Funkcija koja racuna hash kod za dati fajl, kljuc i metod. Koristi se pri kreaciji enkriptovanog fajla, kao i pri dekripciji, u svrhu provere ispravnosti odabranog kljuca/metoda dekripcije. Promenljiva metod ima sledece predvidjene vrednosti: 17 = DES, 71 = triple-DES, 23 = AES 128b, 47 = AES 192b, 61 = AES 256b.

4.4.2.4 `procitajHash()`

```
long long procitajHash (
    char * path )
```

Funkcija koja cita hash kod iz fajla.

Parameters

in	<i>path</i>	Ime fajla iz koga se cita hash kod
----	-------------	------------------------------------

Funkcija se poziva za fajlove u koje je vec upisan hash kod, kada ga zelimo procitati da bismo mogli proveriti validnost kljuca/metoda dekripcije. Vraca poslednjih 8 bajtova.

4.4.2.5 `procitajINFO()`

```
void procitajINFO (
    char * path,
    char * ime,
    int * velicina,
    int * pocetakFajla )
```

Funkcija cita ime i velicinu originala iz enkriptovanog fajla.

Parameters

in	<i>path</i>	Ime enkriptovanog fajla iz koga se cita
out	<i>ime</i>	Ime originala koje se cita iz datog fajla
out	<i>velicina</i>	Velicina originalnog fajla, cita se iz datog fajla
out	<i>pocetakFajla</i>	Duzina informacija upisanih u enkriptovani fajl, koje se trebaju ignorisati pri dekripciji

Funkcija otvara fajl sa imenom *path*, i iz njega učitava dva stringa. Prvi string se direktno smesta u *ime*, a drugi se prvo pretvara u int, a zatim svesta u *velicina*. Duzine dva stringa se sabiraju, i nakon uracunavanja dodatnih karaktera, dobijena vrednost se smesta u *pocetakFajla*, oznacavajuci broj bajtova koji se preskace pri dekripciji.

4.4.2.6 stringHash()

```
int stringHash (
    char * path,
    char fajlVecImaHash )
```

Funkcija koja racuna hash za enkriptovani text.

Parameters

in	<i>path</i>	Ime enkriptovanog fajla za koji se racuna hash njegovog texta
in	<i>fajlVecImaHash</i>	Ima vrednost 1 ako prilozeni fajl vec ima upisan kod na kraju, vrednost 0 u suprotnom

Funkcija koja racuna int vrednost u zavisnosti od enkriptovanog texta datog fajla. Ova vrednosti se kasnije pridruzuju vrednost kljuka i metoda enkripcija i tako se formira hash kod. Funkcija koristi svaki bajt datog fajla, zato moze biti spora za vece fajlove. Da bismo ovu manu sto vise ublazili koriscene su samo bitske operacije.

4.4.2.7 upisiHash()

```
void upisiHash (
    char * path,
    long long hes )
```

Upisuje hash kod u fajl.

Parameters

in	<i>path</i>	Ime fajla u koji se upisuje kod
in	<i>hes</i>	Vrednost koja se upisuje na kraj fajla

Funkcija otvara fajl metodom "ab" i upisuje vrednost *hes* na kraj.

4.5 C:/Users/edvin/Desktop/Attack at Dawn/keydecode.h File Reference

Sve funkcije za konvertovanje kljuka Header file koji sadrzi sve funkcije vezane za konverzije kljuka iz jednog oblika u drugi.


```
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <stdio.h>
```

Functions

- unsigned char * [hex2key](#) (char *[key](#))

4.5.1 Detailed Description

Sve funkcije za konvertovanje ključa Header file koji sadrži sve funkcije vezane za konverzije ključa iz jednog oblika u drugi.

Author

Edvin Maid

4.5.2 Function Documentation

4.5.2.1 hex2key()

```
unsigned char* hex2key (
    char * key )
```

Konvertuje niz heksadecimalnih karaktera u ključ

Parameters

<i>key</i>	Ključ u obliku niza karaktera 0-9 ili A-F
------------	---

Returns

Niz karaktera koji odgovaraju ključu za AES I DES algoritme Uzima niz karaktera koji predstavljaju heksadecimalne cifre i pretvara ih u niz karaktera koji odgovara ključu

4.6 C:/Users/edvin/Desktop/Attack at Dawn/main.c File Reference

Funkcije vezane za učitavanje GUI elemenata i njihovo korišćenje zajedno sa main funkcijom.

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

```
#include <ctype.h>
#include "DES.h"
#include "aes.h"
#include "aadcmd.h"
#include "menus.h"
#include "hash.h"
#include "keydecode.h"
```

Data Structures

- struct [key](#)

struktura koja služi za skladištenje ključa u hexadecimalnoj formi zajedno sa imenom

- struct [couple](#)

pomoćna struktura za povezivanje kolona files keys i algorithm u encryption decryption podmeniju

Macros

- #define **MAX_KEY_LEN** 65
- #define **MAX_KEY_NAME_LEN** 64
- #define **READ_BLOCK_SIZE** 10
- #define **FOR_ALL_SELECTED**(C, F, temp) for(F = C->first_selected;F;temp = F->next_select,[remove_selected_field](#)(C,F),F=temp->next_select)

Typedefs

- typedef enum [algorithm](#) **Algorithm**

enum koriscen da razlikuje algoritme koji ce biti primenjivani na neki fajl

- typedef struct [key](#) **KEY**

struktura koja služi za skladištenje ključa u hexadecimalnoj formi zajedno sa imenom

- typedef struct [couple](#) **COUPLE**

pomoćna struktura za povezivanje kolona files keys i algorithm u encryption decryption podmeniju

Enumerations

- enum [algorithm](#) {
NAA, AES, DES, TRIPLE_DES,
AES_DECRYPT, DES_DECRYPT, TRIPLE_DES_DECRYPT }

enum koriscen da razlikuje algoritme koji ce biti primenjivani na neki fajl

Functions

- void `load ()`
Funkcija koja loaduje sve potrebne elemente i povezuje ih u GUI.
- void `update ()`
Funkcija koja se poziva u svakoj iteraciji programske petlje.
- char * `alg2string (Algorithm a)`
Konvertuje odgovarajuci enum algoritma u njegov reprezentativni string.
- void `key_manager_func ()`
Funkcija koja definise akciju klika na dugme key manager u glavnom meniju Funkcija koja definise akciju klika na dugme key manager u glavnom meniju.
- void `enc_dec_func ()`
Funkcija koja definise akciju klika na dugme encryption decryption menu u glavnom meniju Funkcija koja definise akciju klika na dugme encryption decryption menu u glavnom meniju.
- void `exit_func ()`
Funkcija koja definise akciju klika na dugme exit u glavnom meniju Funkcija koja definise akciju klika na dugme exit u glavnom meniju.
- void `km_right_func ()`
funkcija koja definise akciju klika na dugme >> u key manager podmeniju funkcija koja definise akciju klika na dugme >> u key manager podmeniju
- void `km_left_func ()`
funkcija koja definise akciju klika na dugme << u key manager podmeniju funkcija koja definise akciju klika na dugme << u key manager podmeniju
- void `km_add_func ()`
funkcija koja definise akciju klika na dugme add u key manager podmeniju funkcija koja definise akciju klika na dugme add u key manager podmeniju
- void `km_remv_func ()`
funkcija koja definise akciju klika na dugme remove u key manager podmeniju funkcija koja definise akciju klika na dugme remove u key manager podmeniju
- void `km_edit_func ()`
funkcija koja definise akciju klika na dugme name u key manager podmeniju funkcija koja definise akciju klika na dugme name u key manager podmeniju
- void `km_clear_func ()`
funkcija koja definise akciju klika na dugme clear u key manager podmeniju funkcija koja definise akciju klika na dugme clear u key manager podmeniju
- void `km_load_func ()`
funkcija koja definise akciju klika na dugme load u key manager podmeniju funkcija koja definise akciju klika na dugme load u key manager podmeniju
- void `km_save_func ()`
funkcija koja definise akciju klika na dugme save u key manager podmeniju funkcija koja definise akciju klika na dugme save u key manager podmeniju
- void `km_back_func ()`
funkcija koja definise akciju klika na dugme back u key manager podmeniju funkcija koja definise akciju klika na dugme back u key manager podmeniju
- void `ed_add_func ()`
funkcija koja definise akciju klika na dugme add u encryption decryption podmeniju funkcija koja definise akciju klika na dugme add u encryption decryption podmeniju
- void `ed_remv_func ()`
funkcija koja definise akciju klika na dugme remove u encryption decryption podmeniju funkcija koja definise akciju klika na dugme remove u encryption decryption podmeniju
- void `ed_addf_func ()`
funkcija koja definise akciju klika na dugme add from file u encryption decryption podmeniju funkcija koja definise akciju klika na dugme add from file u encryption decryption podmeniju
- void `ed_solve_func ()`

- funkcija koja definise akciju klika na dugme solve u encryption decryption podmeniju funkcija koja definise akciju klika na dugme solve u encryption decryption podmeniju*
- void `ed_run_func` ()
 - funkcija koja definise akciju klika na dugme run u encryption decryption podmeniju funkcija koja definise akciju klika na dugme run u encryption decryption podmeniju*
- void `ed_dest_func` ()
 - funkcija koja definise akciju klika na dugme back u encryption decryption podmeniju funkcija koja definise akciju klika na dugme back u encryption decryption podmeniju*
- void `ed_back_func` ()
- void `ed_swap` ()
 - funkcija koja definise akciju klika na polje is kolone keys u encryption decryption podmeniju funkcija koja definise akciju klika na polje is kolone keys u encryption decryption podmeniju*
- void `ed_pick_alg` ()
 - funkcija koja definise akciju klika na polje is kolone algorithm u encryption decryption podmeniju funkcija koja definise akciju klika na polje is kolone algorithm u encryption decryption podmeniju*
- void `gen_aux` ()
 - funkcija koja se poziva svaki put kada se pokrene encryption decryption podmeni koja prepisuje kljucve u skrivenu kolonu funkcija koja se poziva svaki put kada se pokrene encryption decryption podmeni koja prepisuje kljucve u skrivenu kolonu*
- void `aux_choose` ()
 - funkcija koja definise akciju klika na polje is skrivene kolone keys u encryption decryption podmeniju funkcija koja definise akciju klika na polje is skrivene kolone keys u encryption decryption podmeniju*
- int `main` (int argc, char **argv)
- `Algorithm shift_alg` (`Algorithm` a, int len)

4.6.1 Detailed Description

Funkcije vezane za učitavanje GUI elemenata i njihovo koriscenje zajedno sa main funkcijom.

Author

Edvin Maid Sve funkcije potrebne za pozivanje aplikacije sastavljenje i učitavanje menija i pokretanje algoritama dekripcije i enkripcije

4.6.2 Function Documentation

4.6.2.1 alg2string()

```
char * alg2string (
    Algorithm a )
```

Konvertuje odgovarajuci enum algoritma u njegov reprezentativni string.

Parameters

<code>a</code>	algoritam koji se konvertuje
----------------	------------------------------

Returns

niz karaktera reprezentativnog string za algoritam a

4.6.2.2 ed_dest_func()

```
void ed_dest_func ( )
```

funkcija koja definise akciju klika na dugme back u encryption decryption podmeniju funkcija koja definise akciju klika na dugme back u encryption decryption podmeniju

funkcija koja definise akciju klika na dugme Destination u encryption decryption podmeniju funkcija koja definise akciju klika na dugme Destination u encryption decryption podmeniju

4.7 C:/Users/edvin/Desktop/Attack at Dawn/menus.h File Reference

Sve funkcije vezane za iscrtavanje i odrzavanje menija Header fajl koji sadrzi sve funkcije koje programer moze da koristi za kreiranje menija i rasporedjivanje GUI elemenata kao i za unos od strane korisnika Oslanja se na biblioteku PDCURSES.

```
#include <curses.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <wchar.h>
```

Data Structures

- struct [field](#)

Struktura koja definise sve potrebne informacije za polje.

- struct [column](#)

Struktura koja definise sve potrebne informacije za kolonu Kolona se sastoji od niza vertikalno rasporedjenih polja.

- struct [menu](#)

Struktura koja definise sve potrebne informacije za Meni Meni se sastoji iz horizontalno postavljenih susednih kolona.

- struct [prog](#)

Struktura koja definise sve potrebne informacije za sam program koji se pokrece.

Macros

- `#define MAX_MENUS 20`
- `#define MAX_COLUMNS 20`
- `#define MAX_NAME_LEN 70`
- `#define MAX_FIELD_NAME_LEN 8`
- `#define MAX_SELECT 2000`
- `#define MAX_FIELD_STRING_LEN 70`
- `#define MAX_ERROR_MESSAGE_LEN 100`
- `#define MAX_INPUT_LEN 500`
- `#define MENUS_ERR 0`
- `#define MENUS_OK 1`
- `#define HEADER_WIDTH 0`
- `#define edvin endwin`

Typedefs

- `typedef enum menus_error MENUS_ERROR`
- `typedef struct field FIELD`
Struktura koja definise sve potrebne informacije za polje.
- `typedef struct column COLUMN`
Struktura koja definise sve potrebne informacije za kolonu Kolona se sastoji od niza vertikalno raspoređenih polja.
- `typedef struct menu MENU`
Struktura koja definise sve potrebne informacije za Meni Meni se sastoji iz horizontalno postavljenih susednih kolona.
- `typedef struct prog PROG`
Struktura koja definise sve potrebne informacije za sam program koji se pokrece.

Enumerations

- `enum menus_error {`
`M_ERR_NO_ERROR, M_ERR_NO_INITIALIZED_PROG, M_ERR_NULL_PONITER, M_ERR_OVERFL↵
OW,
M_ERR_UNDERFLOW, M_ERR_ALLOC, M_ERR_FIELD_POSITION_FILLED, M_ERR_PROG_ALRE↵
ADY_RUNNING,
M_ERR_PROG_ALREADY_ACTIVE, M_ERR_SCREEN_DIM }`

Functions

- `int init_prog ()`
funkcija koja inicijalizuje program Kreira novi program koji skladišti podatke vezane za menije takodje i vrsi potrebnu inicijalizaciju za biblioteku pdcurses
- `void set_update (void(*update)(void))`
postavlja funkciju koja ce da se poziva u svakoj iteraciji programske petlje
- `int run_prog (MENU *startMenu)`
Pokrece program koji je prethodno bio inicijalizovan.
- `void stop_prog ()`
izbacuje program iz programske petlje
- `int end_prog ()`
Oslobadja memoriju Oslobadja sve podatke vezane za menije.
- `MENU * get_active_menu ()`
Vraca trenutno aktivni meni.

- void `nothing` (void *)
Ne radi nista bas nista.
- void `nothing2` (void)
Ne radi nista bas bas nista.
- `MENU *` `get_menu` (char *menuName)
vraca meni sa zadatim imenom
- `MENU *` `new_menu` (char *menuName)
kreira novi meni
- int `update_menu` (`MENU *`M)
Funkcija koja se pokrece u svakoj iteraciji programskog ciklusa.
- int `add_column` (`MENU *`M, `COLUMN *`C)
Dodaje kolonu u odgovarajuci meni.
- int `prev_menu` ()
Vraca aktivni meni unazad.
- int `next_menu` (`MENU *`M)
Postavlja sledeci meni na stek.
- void `print_menu` (`MENU *`M)
Iscrtava meni.
- `COLUMN *` `get_column` (`MENU *`M, char *columnName)
Dohvata kolonu sa zadatim imenom.
- `COLUMN *` `new_column` (char *columnName, char *columnTitle, int width)
Kreira novu kolonu.
- int `add_field` (`COLUMN *`C, `FIELD *`F, int position)
Dodaje polje u zadatu kolonu.
- int `print_column` (`COLUMN *`C)
Ispisuje kolonu na ekran.
- void `set_border` (`COLUMN *`C, int hasBorder, int a, int b)
Postavlja okvir zadate kolone.
- void `lockColumn` (`COLUMN *`C, int is_locked, int lock_pos)
Zakljucava poziciju kolone.
- void `compact` (`COLUMN *`C)
Sabija sva polja kolone tako da nema razmaka.
- `FIELD *` `get_field` (`COLUMN *`C, char *fieldName)
Pronalazi polje sa zadatim imenom.
- `FIELD *` `new_field` (void *x, void(*f)(void), void(*free_func)(void *), char *fieldName, char *fieldString)
Kreira novo polje.
- void `set_field_string` (`FIELD *`F, char *fieldString)
Menja vrednost niza karaktera koji se ispisuje.
- void `execute` (`FIELD *`F)
- void `select` ()
Obelezava polje na kojem je kursor Funkcija koja obelezava polje nad kojim je kursor u aktivnom meniju i dodaje u listu obelezanih polja. Ova funkcija se uzima kao podrazumevana ako je drugi parametar funkcije `new_field()` jednak NULL.
- `FIELD *` `get_f` (`COLUMN *`C, int index)
Vraca polje sa zadatim indeksom u listi polja.
- void `remove_field` (`COLUMN *`C, `FIELD *`F)
Uklanja zadato polje iz liste polja kolone C.
- void `remove_selected_field` (`COLUMN *`C, `FIELD *`F)
Polje varca u neobelezeno stanje.
- int `getPos` (`COLUMN *`C, `FIELD *`F)
Vraca poziciju polja F ukoloni C.

- void `print_field` (`COLUMN *C`, `FIELD *F`, int pos)
Iscrtava polje F.
- void `print_title` (`COLUMN *C`, char *s)
Ispisuje naslov zadatke kolone.
- void `free_field` (`FIELD *F`)
Oslobadja memoriju vezanu za polje.
- int `cursor_up` ()
Pomera kursor gore.
- int `cursor_down` ()
Pomera kursor dole.
- int `cursor_left` ()
Pomera kursor u susednu levu kolonu.
- int `cursor_right` ()
Pomera kursor u susednu desnu kolonu.
- void `input_box` (int height, int width, char *title, char *message, char *input)
Pokrece primitivni input-box pop up za unos niza karaktera.
- void `message_box` (int height, int width, char *title, char *message)
Pokrece primitivni pop up za prikazivanje poruke.

4.7.1 Detailed Description

Sve funkcije vezane za iscrtavanje i odrzavanje menija Header fajl koji sadrzi sve funkcije koje programer moze da koristi za kreiranje menija i rasporedjivanje GUI elemenata kao i za unos od strane korisnika Oslanja se na biblioteku PDCURSES.

Author

Edvin Maid

4.7.2 Function Documentation

4.7.2.1 `add_column()`

```
int add_column (
    MENU * M,
    COLUMN * C )
```

Dodaje kolonu u odgovarajuci meni.

Parameters

<i>M</i>	Meni u koji zelimo da dodamo kolonu
<i>C</i>	Kolona koju zelimo da dodamu u odgovarajuci meni

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvršeno ili `MENUS_ERR` ukoliko je doslo do greske

4.7.2.2 add_field()

```
int add_field (
    COLUMN * C,
    FIELD * F,
    int position )
```

Dodaje polje u zadatu kolonu.

Parameters

<i>C</i>	Kolona u koju dodaje polje
<i>F</i>	Polje koje zelimo da dodamo
<i>position</i>	Pozicija odnosno broj mesta od pocetka kolone

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvršeno ili `MENUS_ERR` ukoliko je doslo do greske Dodaje zadato polje u zadatu kolonu na neku poziciju. Ako na primer dodamo jedno polje na poziciju 1 a drugo na poziciju 4 na ekranu ce se ispisati:

polje1

polje2

Ako prosledimo `position < 0` onda ce samo dodati polje na kraj

4.7.2.3 compact()

```
void compact (
    COLUMN * C )
```

Sabija sva polja kolone tako da nema razmaka.

Parameters

<i>C</i>	Kolona koju zelimo da sabijemo Pozicije kolone postaju zbijene na primer ako kolona poseduje polje sa pozicijom 1 i polje sa pozicijom 4 ispisivace se: polje1 polje2
----------	---

bez razmaka izmedju redova

4.7.2.4 cursor_down()

```
int cursor_down ( )
```

Pomera kursor dole.

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvršeno ili `MENUS_ERR` ukoliko je doslo do greske Pomera kursor na ekranu za jedno polje dole i ukoliko naidje na gornju granicu kolone vrši "scroll"

4.7.2.5 `cursor_left()`

```
int cursor_left ( )
```

Pomera kursor u susednu levu kolonu.

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvršeno ili `MENUS_ERR` ukoliko je doslo do greske Pomera kursor u prvo vidljivo polje kolone koja se nalazi levo od trenutno kolone, a ukoliko ne postoji leva kolona koja sadrzi bar jedno polje onda kursor ostaje u istoj koloni

4.7.2.6 `cursor_right()`

```
int cursor_right ( )
```

Pomera kursor u susednu desnu kolonu.

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvršeno ili `MENUS_ERR` ukoliko je doslo do greske Pomera kursor u prvo vidljivo polje kolone koja se nalazi desno od trenutno kolone, a ukoliko ne postoji desna kolona koja sadrzi bar jedno polje onda kursor ostaje u istoj koloni

4.7.2.7 `cursor_up()`

```
int cursor_up ( )
```

Pomera kursor gore.

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvršeno ili `MENUS_ERR` ukoliko je doslo do greske Pomera kursor na ekranu za jedno polje gore i ukoliko naidje na gornju granicu kolone vrši "scroll"

4.7.2.8 end_prog()

```
int end_prog ( )
```

Oslobadja memoriju Oslobadja sve podatke vezane za menije.

Returns

definiciju MENUS_OK ukoliko je uspesno izvršeno ili MENUS_ERR ukoliko je doslo do greske

4.7.2.9 free_field()

```
void free_field (
    FIELD * F )
```

Oslobadja memoriju vezanu za polje.

Parameters

<i>F</i>	Briše memoriju polja F kao i poziva zadatu funkciju za brisanje memorije asocirane sa poljem F
----------	--

4.7.2.10 get_active_menu()

```
MENU* get_active_menu ( )
```

Vraca trenutno aktivni meni.

Returns

Pokazivac na strukturu MENU koja odgovara meniju koji se trenutno nalazi na ekranu

4.7.2.11 get_column()

```
COLUMN* get_column (
    MENU * M,
    char * columnName )
```

Dohvata kolonu sa zadatim imenom.

Parameters

<i>M</i>	Meni u kojoj se nalazi trazena kolona
<i>columnName</i>	niz karaktera koji odgovara imenu kolone

Returns

Pokazivac na trazenu kolonu ili NULL ukoliko nije nadjena Prolazi kroz niz svih kolona u zadatom meniju i trazi onu sa zadatim imenom

4.7.2.12 get_f()

```
FIELD* get_f (
    COLUMN * C,
    int index )
```

Vraca polje sa zadatim indeksom u listi polja.

Parameters

<i>C</i>	Kolona u kojoj tazimo polje
<i>index</i>	indeks polja koje trazimo pocevsi od 0

Returns

pokazivac na polje sa zadatim indeksom Prolazi kroz listu polja kolone C dok ne pronadje polje sa zadatim indeksom index

4.7.2.13 get_field()

```
FIELD* get_field (
    COLUMN * C,
    char * fieldName )
```

Pronalazi polje sa zadatim imenom.

Parameters

<i>C</i>	Kolona u kojoj se nalazi polje
<i>fieldName</i>	Ime trazenog polja

Returns

Pokazivac na polje sa zadatim imenom ili NULL ako nije nadjen Prolazi kroz listu polja u kolonu C i poredi sa fieldName dok ne nadje odgovarajuće polje

4.7.2.14 get_menu()

```
MENU* get_menu (
    char * menuName )
```

vraca meni sa zadatim imenom

Parameters

<i>menuName</i>	niz karaktera koji odgovara meniju
-----------------	------------------------------------

Returns

pokazivac na meni sa zadatim imenom ili NULL ukoliko nije nadjen Ova funkcija prolazi kroz niz menija ucitanih u program i pronalazi odgovarajuci

4.7.2.15 getPos()

```
int getPos (
    COLUMN * C,
    FIELD * F )
```

Vraca poziciju polja F ukoloni C.

Parameters

<i>C</i>	Kolona u kojoj se nalazi F
<i>F</i>	Polje ciju poziciju trazimo

Returns

pozicija polja F Vraca nam vrednost pozicije polja F

4.7.2.16 init_prog()

```
int init_prog ( )
```

funkcija koja inicijalizuje program Kreira novi program koji skladisti podatke vezane za menije takodje i vrsi potrebnu inicijalizaciju za biblioteku pdcurses

Returns

4.7.2.17 input_box()

```
void input_box (
    int height,
    int width,
    char * title,
    char * message,
    char * input )
```

Pokrece primitivni input-box pop up za unos niza karaktera.

Parameters

<i>height</i>	Visina prozora pop up-a u broju karaktera
<i>width</i>	Sirina prozora pop up-a u broju karaktera
<i>title</i>	Niz karaktera koji ce biti ispisan na vrhu prozora
<i>message</i>	Niz karaktera koji ce biti ispisan u prozoru
<i>input</i>	niz karaktera u kojem ce se smestiti korisnicki unos Funkcija pokrece prozor u kojem ce biti ispisana poruka zajedno sa naslovom i jednim poljem za unos sa najvise MAX_INPUT_LEN karaktera

4.7.2.18 lockColumn()

```
void lockColumn (
    COLUMN * C,
    int is_locked,
    int lock_pos )
```

Zakljucava poziciju kolone.

Parameters

<i>C</i>	Kolona koju zelimo da zakljucamo
<i>is_locked</i>	1 ako zelimo da zakljucamo 0 ako zelimo da otkljucamo
<i>lock_pos</i>	pozicija na kojoj zelimo da zakljucamo Pomeranjem kursora kada se naidje na ivice kolone ne vrsi se "scroll" a na vrhu kolone iscrtane na ekranu se nalazi polje sa pozicijom lock_pos

4.7.2.19 message_box()

```
void message_box (
    int height,
    int width,
    char * title,
    char * message )
```

Pokrece primitivni pop up za prikazivanje poruke.

Parameters

<i>height</i>	Visina prozora pop up-a u broju karaktera
<i>width</i>	Sirina prozora pop up-a u broju karaktera
<i>title</i>	Niz karaktera koji ce biti ispisan na vrhu prozora
<i>message</i>	Niz karaktera koji ce biti ispisan u prozoru Funkcija pokrece prozor u kojem ce biti ispisana poruka zajedno sa naslovom

4.7.2.20 new_column()

```
COLUMN* new_column (
    char * columnName,
    char * columnTitle,
    int width )
```

Kreira novu kolonu.

Parameters

<i>columnName</i>	Ime koje zelimo da pripisemo novoj koloni
<i>columnTitle</i>	Niz karaktera koji zelimo da se ispisuje u zacelju kolone
<i>width</i>	Sirinu kolone zadatu u broju karaktera

Returns

Pokazivac na novu kolonu

4.7.2.21 new_field()

```
FIELD* new_field (
    void * x,
    void(*) (void) f,
    void(*) (void *) free_func,
    char * fieldName,
    char * fieldString )
```

Kreira novo polje.

Parameters

<i>x</i>	Pokazivac na memoriju koju zelimo da asociramo sa zadatim poljem
<i>f</i>	Pokazivac na funkciju akcije klika na kreirano polje
<i>free_func</i>	funkcija koja oslobadja memoriju zadatu parametrom x
<i>fieldName</i>	MAX_FIELD_NAME_LEN karaktera koji figurisu ime kreiranog polja
<i>fieldString</i>	MAX_FIELD_STRING_LEN karaktera koje zelimo da ispisemo na polju

Returns

Pokazivac na kreirano polje Kreira polje sa zadatim nizom karaktera za ispis, zadatim imenom, nekom memorijom sa kojom se asocira i funkcijama za brisanje te memorije kao i za akcije prilikom klika na polje Ukoliko je parametar f jednak NULL podrazumeva se funkcija [select\(\)](#) Ukoliko je parametar free_func jednak NULL podrazumeva se funkcija free()

4.7.2.22 new_menu()

```
MENU* new_menu (
    char * menuName )
```

kreira novi meni

Parameters

<i>menuName</i>	Naziv menija koji ce se posle koristiti da se njemu pristupi
-----------------	--

Returns

Pokazivac na kreirani meni

4.7.2.23 next_menu()

```
int next_menu (
    MENU * M )
```

Postavlja sledeci meni na stek.

Parameters

<i>M</i>	meni koji zelimo da postane aktivan
----------	-------------------------------------

Returns

definiciju MENUS_OK ukoliko je uspesno izvrшено ili MENUS_ERR ukoliko je doslo do greske Postavlja zadati meni na stek i proglašava ga aktivnim menijem

4.7.2.24 nothing()

```
void nothing (
    void * )
```

Ne radi nista bas nista.

Parameters

<i>Pokazivac</i>	na nesto
------------------	----------

4.7.2.25 prev_menu()

```
int prev_menu ( )
```

Vraca aktivni meni unazad.

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvrшено ili `MENUS_ERR` ukoliko je doslo do greske Uzima poslenji meni sa steka menija i postavlja ga kao aktivni meni

4.7.2.26 print_column()

```
int print_column (
    COLUMN * C )
```

Ispisuje kolonu na ekran.

Parameters

<i>C</i>	Kolona koju zelimo da ispisemo na ekran
----------	---

Returns

definiciju `MENUS_OK` ukoliko je uspesno izvrшено ili `MENUS_ERR` ukoliko je doslo do greske Poziva se u [print_menu\(\)](#) i iscrtava sva polje i okvir zadate kolone na odgovarajucoj poziciji

4.7.2.27 print_field()

```
void print_field (
    COLUMN * C,
    FIELD * F,
    int pos )
```

Iscrtava polje F.

Parameters

<i>C</i>	Kolona u kojoj se nalazi polje F
<i>F</i>	polje ciji field_string zelimo da ispisemo
<i>pos</i>	poziciju od vrha iscrtane kolone na kojoj zelimo da iscrtamo polje Iscrtava polje iz kolone C na poziciji pos od vrha kolone

4.7.2.28 print_menu()

```
void print_menu (
    MENU * M )
```

Iscrtava meni.

Parameters

<i>M</i>	Meni koji zelimo da iscrtamo Iscrtava sve elemente prosledjenog menija na ekran poziva se u update_menu
----------	---

4.7.2.29 print_title()

```
void print_title (
    COLUMN * C,
    char * s )
```

Ispisuje naslov zadatke kolone.

Parameters

<i>C</i>	Kolona ciji naslov ispisuje
<i>s</i>	Niz karaktera koji zelimo da ispisemo kao naslov Ispisuje niz karaktera s u zaglavlju kolone C

4.7.2.30 remove_field()

```
void remove_field (
    COLUMN * C,
    FIELD * F )
```

Uklanja zadato polje iz liste polja kolone C.

Parameters

<i>C</i>	Kolona iz koje zelimo da uklonimo polje F
<i>F</i>	Polje koje zelimo da uklonimo iz kolone C Funkcija prolazi kroz listu polja u koloni C i pronalazi F. Ukoliko ne pronadje odgovarajuće polje nista se ne menja. Pronadjeno F je zatim izbaceno iz liste.
	Generated by Doxygen

4.7.2.31 `remove_selected_field()`

```
void remove_selected_field (
    COLUMN * C,
    FIELD * F )
```

Polje varca u neobeleženo stanje.

Parameters

<i>C</i>	Kolona u kojoj se nalazi obeleženo polje
<i>F</i>	Polje cije stanje zelimo da pretvorimo u neobeleženo Funkcija ukoliko je F obeleženo ga pronalazi i pretvara u neobeleženo polje

4.7.2.32 `run_prog()`

```
int run_prog (
    MENU * startMenu )
```

Pokrece program koji je prethodno bio inicijalizovan.

Parameters

<i>startMenu</i>	meni iz kojeg program pocinje
------------------	-------------------------------

Returns

definiciju MENUS_OK ukoliko je uspesno izvršeno ili MENUS_ERR ukoliko je doslo do greske

4.7.2.33 `set_border()`

```
void set_border (
    COLUMN * C,
    int hasBorder,
    int a,
    int b )
```

Postavlja okvir zadate kolone.

Parameters

<i>C</i>	Kolone kojoj zelimo da dodamo okvir
<i>hasBorder</i>	1 ako zelimo okvir 0 ako zelimo bez okvira
<i>a</i>	karakter okvira svuda sem coskova, 0 za crticu
<i>b</i>	karakter u coskovima okvira, 0 za prave uglove Postavlja kolonu tako da iscrtava svoj okvir u <code>print_column()</code>

4.7.2.34 set_field_string()

```
void set_field_string (
    FIELD * F,
    char * fieldString )
```

Menja vrednost niza karaktera koji se ispisuje.

Parameters

<i>F</i>	Polje ciji ispis zelimo da promenimo
<i>fieldString</i>	niz karaktera koji ce se ispisivati Kopira zadati niz karaktera u polje i ispisuje

4.7.2.35 set_update()

```
void set_update (
    void(*) (void) update )
```

postavlja funkciju koja ce da se poziva u svakoj iteraciji progamske petlje

Parameters

<i>update</i>	funkcija koju ce program pozivati Kao argument se prosledjuje funkcija za koju programer zeli da se izvršava u svakoj iteraciji programske petlje
---------------	---

4.7.2.36 update_menu()

```
int update_menu (
    MENU * M )
```

Funkcija koja se pokrece u svakoj iteraciji programskog ciklusa.

Parameters

<i>M</i>	Meni koji ce se iscrtati i azurirati
----------	--------------------------------------

Returns

definiciju MENUS_OK ukoliko je uspesno izvršeno ili MENUS_ERR ukoliko je doslo do greske Iscrtava prosledjeni meni poziva funkciju update prosledjenu kao argument u funkciji set_update i zaustavlja program i ceka unos od strane korisnika

Index

aadcmd.h
 aes_cmd, 10
 batch_mode, 10
 des_cmd, 11
 do_batch, 11
 interpret, 11
 open_log, 12
 print_to_log, 12
 start_cmd_mode, 12
 triple_des_cmd, 13
add_column
 menus.h, 34
add_field
 menus.h, 35
AddKey
 aes.h, 14
aes.h
 AddKey, 14
 Aes_Cipher_Block, 14
 Aes_Cipher_File, 15
 Aes_Decipher_Block, 16
 Aes_Decipher_File, 16
 ime_provera, 17
 InverseMixColumns, 17
 InverseShiftRows, 17
 InverseSubBytes, 18
 KeyExpansion, 18
 MixColumns, 19
 ShiftRows, 19
 SubBytes, 19
 SubWord, 20
Aes_Cipher_Block
 aes.h, 14
Aes_Cipher_File
 aes.h, 15
Aes_Decipher_Block
 aes.h, 16
Aes_Decipher_File
 aes.h, 16
aes_cmd
 aadcmd.h, 10
alg2string
 main.c, 30

batch_mode
 aadcmd.h, 10
blok, 5

C:/Users/edvin/Desktop/Attack at Dawn/DES.h, 20
C:/Users/edvin/Desktop/Attack at Dawn/aadcmd.h, 9
C:/Users/edvin/Desktop/Attack at Dawn/aes.h, 13
C:/Users/edvin/Desktop/Attack at Dawn/hash.h, 23
C:/Users/edvin/Desktop/Attack at Dawn/keydecode.h, 26
C:/Users/edvin/Desktop/Attack at Dawn/main.c, 27
C:/Users/edvin/Desktop/Attack at Dawn/menus.h, 31
column, 5
compact
 menus.h, 35
couple, 6
cursor_down
 menus.h, 35
cursor_left
 menus.h, 36
cursor_right
 menus.h, 36
cursor_up
 menus.h, 36

DES.h
 DES_decrypt_file, 21
 DES_encrypt_file, 21
 triple_DES_decrypt_file, 21
 triple_DES_encrypt_file, 23
DES_decrypt_file
 DES.h, 21
DES_encrypt_file
 DES.h, 21
des_cmd
 aadcmd.h, 11
do_batch
 aadcmd.h, 11
Dodaj_ime_i_velicinu
 hash.h, 24

ed_dest_func
 main.c, 31
end_prog
 menus.h, 36

field, 6
free_field
 menus.h, 37

get_active_menu
 menus.h, 37
get_column
 menus.h, 37
get_f
 menus.h, 38

- get_field
 - menus.h, [38](#)
- get_menu
 - menus.h, [38](#)
- getPos
 - menus.h, [39](#)
- hash.h
 - Dodaj_ime_i_velicinu, [24](#)
 - isGood, [24](#)
 - mojHash, [25](#)
 - procitajHash, [25](#)
 - procitajINFO, [25](#)
 - stringHash, [26](#)
 - upisiHash, [26](#)
- hex2key
 - keydecode.h, [27](#)
- ime_provera
 - aes.h, [17](#)
- init_prog
 - menus.h, [39](#)
- input_box
 - menus.h, [39](#)
- interpret
 - aadcmd.h, [11](#)
- InverseMixColumns
 - aes.h, [17](#)
- InverseShiftRows
 - aes.h, [17](#)
- InverseSubBytes
 - aes.h, [18](#)
- isGood
 - hash.h, [24](#)
- key, [7](#)
- KeyExpansion
 - aes.h, [18](#)
- keydecode.h
 - hex2key, [27](#)
- lockColumn
 - menus.h, [40](#)
- main.c
 - alg2string, [30](#)
 - ed_dest_func, [31](#)
- menu, [7](#)
- menus.h
 - add_column, [34](#)
 - add_field, [35](#)
 - compact, [35](#)
 - cursor_down, [35](#)
 - cursor_left, [36](#)
 - cursor_right, [36](#)
 - cursor_up, [36](#)
 - end_prog, [36](#)
 - free_field, [37](#)
 - get_active_menu, [37](#)
 - get_column, [37](#)
 - get_f, [38](#)
 - get_field, [38](#)
 - get_menu, [38](#)
 - getPos, [39](#)
 - init_prog, [39](#)
 - input_box, [39](#)
 - lockColumn, [40](#)
 - message_box, [40](#)
 - new_column, [41](#)
 - new_field, [41](#)
 - new_menu, [42](#)
 - next_menu, [42](#)
 - nothing, [42](#)
 - prev_menu, [43](#)
 - print_column, [43](#)
 - print_field, [43](#)
 - print_menu, [44](#)
 - print_title, [44](#)
 - remove_field, [44](#)
 - remove_selected_field, [45](#)
 - run_prog, [45](#)
 - set_border, [45](#)
 - set_field_string, [46](#)
 - set_update, [46](#)
 - update_menu, [46](#)
- message_box
 - menus.h, [40](#)
- MixColumns
 - aes.h, [19](#)
- mojHash
 - hash.h, [25](#)
- new_column
 - menus.h, [41](#)
- new_field
 - menus.h, [41](#)
- new_menu
 - menus.h, [42](#)
- next_menu
 - menus.h, [42](#)
- nothing
 - menus.h, [42](#)
- open_log
 - aadcmd.h, [12](#)
- prev_menu
 - menus.h, [43](#)
- print_column
 - menus.h, [43](#)
- print_field
 - menus.h, [43](#)
- print_menu
 - menus.h, [44](#)
- print_title
 - menus.h, [44](#)
- print_to_log
 - aadcmd.h, [12](#)

procitajHash
 hash.h, [25](#)
procitajINFO
 hash.h, [25](#)
prog, [8](#)

remove_field
 menus.h, [44](#)
remove_selected_field
 menus.h, [45](#)
run_prog
 menus.h, [45](#)

set_border
 menus.h, [45](#)
set_field_string
 menus.h, [46](#)
set_update
 menus.h, [46](#)
ShiftRows
 aes.h, [19](#)
start_cmd_mode
 aadcmd.h, [12](#)
stringHash
 hash.h, [26](#)
SubBytes
 aes.h, [19](#)
SubWord
 aes.h, [20](#)

triple_DES_decrypt_file
 DES.h, [21](#)
triple_DES_encrypt_file
 DES.h, [23](#)
triple_des_cmd
 aadcmd.h, [13](#)

update_menu
 menus.h, [46](#)
upsiHash
 hash.h, [26](#)