My Project

Generated by Doxygen 1.9.4

Chapter 1

File Index

1.1 File List

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

Chapter 2

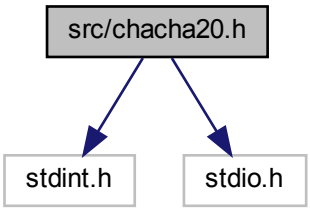
File Documentation

2.1 src/chacha20.h File Reference

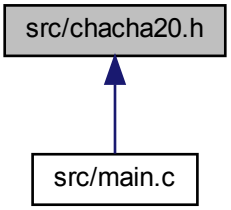
Header file for ChaCha20 stream cipher implementation.

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

Include dependency graph for chacha20.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **ROTL32**(v, n) ((v << n) | (v >> (32 - n)))

  Rotates a 32-bit unsigned integer `v` left by `n` bits.

Functions

- uint8_t * get_key ()

  Prompts the user for a password and returns a 32-byte key.
- uint8_t * get_nonce ()

  Generates a 12-byte nonce using /dev/urandom or fallback to rand().
- uint32_t load32_le (const uint8_t *src)

  Loads a 32-bit unsigned integer from a little-endian byte array.
- void store32_le (uint8_t *dst, uint32_t w)

  Stores a 32-bit unsigned integer into a byte array in little-endian format.
- void quarterround (uint32_t *a, uint32_t *b, uint32_t *c, uint32_t *d)

  Applies the ChaCha20 quarter round function to four 32-bit words.
- void chacha20_block (uint32_t state[16], uint8_t output[64])

  Executes the ChaCha20 block function on a given 512-bit state.
- void poly1305_auth (uint8_t *mac, const uint8_t *msg, size_t msg_len, const uint8_t key[32])

  Generates a Poly1305 authentication tag for the given message.

2.1.1 Detailed Description

Header file for ChaCha20 stream cipher implementation.

Author

Subramanya J subramanyajaradhya@gmail.com

Implements the functions defined in chacha20.h

Author

Subramanya J subramanyajaradhya@gmail.com

This file provides the function declarations and macros needed to use the ChaCha20 encryption block function, along with helper routines for key and nonce handling.

2.1.2 Function Documentation

2.1.2.1 chacha20_block()

```
void chacha20_block (
            uint32_t state[16],
            uint8_t output[64] )
```

Executes the ChaCha20 block function on a given 512-bit state.

The output is 64 bytes (512 bits) of keystream, generated from 10 double-rounds.

Parameters

state	Pointer to the 16-word (64-byte) input state.
output	Pointer to a 64-byte buffer where output will be stored.

2.1.2.2 get_key()

```
uint8_t * get_key ( )
```

Prompts the user for a password and returns a 32-byte key.

The password entered by the user is truncated or zero-padded to 32 bytes.

Returns

Pointer to a heap-allocated 32-byte buffer containing the key. Must be freed by the caller.

2.1.2.3 get_nonce()

```
uint8_t * get_nonce ( )
```

Generates a 12-byte nonce using /dev/urandom or fallback to rand().

Tries to read from `/dev/urandom`; falls back to `rand()` if unavailable.

Returns

Pointer to a heap-allocated 12-byte buffer containing the nonce. Must be freed by the caller.

2.1.2.4 load32_le()

```
uint32_t load32_le (
            const uint8_t * src )
```

Loads a 32-bit unsigned integer from a little-endian byte array.

Parameters

src	Pointer to 4 bytes in memory in little-endian order.

Returns

32-bit unsigned integer.

2.1.2.5 poly1305_auth()

```
void poly1305_auth (
            uint8_t * mac,
            const uint8_t * msg,
            size_t msg_len,
            const uint8_t key[32] )
```

Generates a Poly1305 authentication tag for the given message.

This tag is used for message authentication in the ChaCha20-Poly1305 scheme.

Parameters

mac	Pointer to a 16-byte buffer for the generated tag.
msg	Pointer to the message to be authenticated.
msg_len	Length of the message in bytes.
key	32-byte key for Poly1305.

2.1.2.6 quarterround()

```
void quarterround (
            uint32_t * a,
            uint32_t * b,
            uint32_t * c,
            uint32_t * d )
```

Applies the ChaCha20 quarter round function to four 32-bit words.

This function performs 4 rounds of ARX (Add-Rotate-XOR) operations on the inputs. b ^= (a+d) <<< 7; c ^= (b+a) <<< 9; d ^= (c+b) <<< 13; a ^= (d+c) <<< 18;

2.1.2.7 store32_le()

```
void store32_le (
            uint8_t * dst,
            uint32_t w )
```

Stores a 32-bit unsigned integer into a byte array in little-endian format.

Parameters

dst	Destination pointer to a 4-byte memory location.
w	32-bit unsigned integer to store.

2.2 chacha20.h

Go to the documentation of this file.

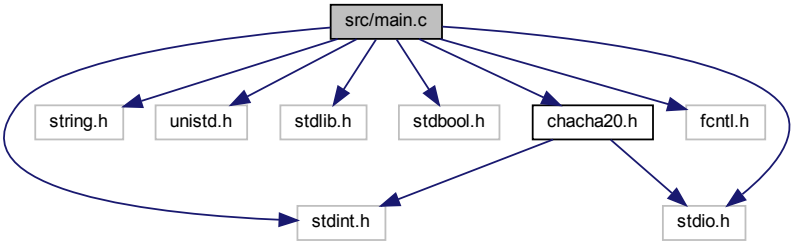
```
1
13 #ifndef CHACHA_20_H
14 #define CHACHA_20_H
15
16 #include <stdint.h>
17 #include <stdio.h>
18 #include <stdint.h>
19
28 uint8_t *
29 get_key();
30
39 uint8_t *
40 get_nonce();
41
48 uint32_t
49 load32_le(const uint8_t *src);
50
57 void
58 store32_le(uint8_t *dst, uint32_t w);
59
63 #define ROTL32(v, n) ((v « n) | (v » (32 - n)))
64
65
75 void quarterround(uint32_t *a, uint32_t *b, uint32_t *c, uint32_t *d);
76
85 void chacha20_block(uint32_t state[16], uint8_t output[64]);
86
97 void poly1305_auth(uint8_t *mac, const uint8_t *msg, size_t msg_len, const uint8_t key[32]);
98
99 #endif /* chacha20.h included */
```

2.3 src/main.c File Reference

Entry point for the ChaCha20 file encryption/decryption utility.

```
#include <stdint.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>
#include <fcntl.h>
#include "chacha20.h"
```

Include dependency graph for main.c:



Functions

- int main (int argc, char **argv)

  Main function to handle encryption/decryption based on command-line arguments.

2.3.1 Detailed Description

Entry point for the ChaCha20 file encryption/decryption utility.

This program encrypts or decrypts a file using the ChaCha20 stream cipher. It reads from an input file, processes 64-byte blocks, and writes to an output file. It uses a password-derived key and a 12-byte nonce.

Usage: ./program <filename> <e|d>

- 'e' to encrypt the file
- 'd' to decrypt a previously encrypted file

Author

Subramanya J subramanyajaradhya@gmail.com

2.3.2 Function Documentation

2.3.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Main function to handle encryption/decryption based on command-line arguments.

The program uses a password-derived key for encryption or decryption of a file. For encryption, it generates a random 12-byte nonce and writes it to the output file. For decryption, it reads the nonce from the input file.

Parameters

argc	Argument count.
argv	Argument vector. argv[1]: input filename argv[2]: 'e' for encryption, 'd' for decryption

Returns

int Exit status code (0 for success, 1 for argument error, or other error codes).

Index