

# Steganography on Vorbis' audio streams

Generated by Doxygen 1.6.1

Mon Aug 30 18:29:37 2010



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	cryptos_config_t Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Member Data Documentation . . . . .	5
3.1.2.1	chd . . . . .	5
3.1.2.2	cipher_algo . . . . .	6
3.1.2.3	default_data_size . . . . .	6
3.1.2.4	emission . . . . .	6
3.1.2.5	hmac . . . . .	6
3.1.2.6	iv . . . . .	6
3.1.2.7	key . . . . .	6
3.1.2.8	max_data . . . . .	6
3.1.2.9	md_algo . . . . .	6
3.1.2.10	md_len . . . . .	6
3.1.2.11	mdhd . . . . .	6
3.1.2.12	packet . . . . .	6
3.2	cryptos_key_t Struct Reference . . . . .	7
3.2.1	Detailed Description . . . . .	7
3.2.2	Member Data Documentation . . . . .	7
3.2.2.1	key . . . . .	7
3.2.2.2	length . . . . .	7
3.3	cryptos_protocol_buffer Struct Reference . . . . .	8
3.3.1	Detailed Description . . . . .	8

3.4	<a href="#">cryptos_protocol_buffer_t Struct Reference</a>	9
3.4.1	<a href="#">Member Data Documentation</a>	9
3.4.1.1	<a href="#">buffer</a>	9
3.4.1.2	<a href="#">buffer_size</a>	9
3.4.1.3	<a href="#">buffer_used</a>	9
3.4.1.4	<a href="#">fd</a>	9
3.4.1.5	<a href="#">offset</a>	9
3.5	<a href="#">iss_cfg_t Struct Reference</a>	10
3.5.1	<a href="#">Member Data Documentation</a>	10
3.5.1.1	<a href="#">alpha</a>	10
3.5.1.2	<a href="#">lambda</a>	10
3.5.1.3	<a href="#">sigma</a>	10
3.5.1.4	<a href="#">u</a>	10
3.5.1.5	<a href="#">u_norm</a>	10
3.6	<a href="#">prng_t Struct Reference</a>	11
3.6.1	<a href="#">Detailed Description</a>	11
3.6.2	<a href="#">Member Data Documentation</a>	11
3.6.2.1	<a href="#">iters</a>	11
3.6.2.2	<a href="#">seed</a>	11
3.7	<a href="#">steganos_key_t Struct Reference</a>	12
3.7.1	<a href="#">Detailed Description</a>	12
3.7.2	<a href="#">Member Data Documentation</a>	12
3.7.2.1	<a href="#">key</a>	12
3.7.2.2	<a href="#">length</a>	12
3.8	<a href="#">steganos_state_t Struct Reference</a>	13
3.8.1	<a href="#">Detailed Description</a>	13
3.8.2	<a href="#">Member Data Documentation</a>	14
3.8.2.1	<a href="#">aligned</a>	14
3.8.2.2	<a href="#">da</a>	14
3.8.2.3	<a href="#">desync</a>	14
3.8.2.4	<a href="#">desynchronize</a>	14
3.8.2.5	<a href="#">hide</a>	14
3.8.2.6	<a href="#">hide_method</a>	14
3.8.2.7	<a href="#">hiding_key</a>	14
3.8.2.8	<a href="#">iters</a>	14
3.8.2.9	<a href="#">master_key</a>	14

3.8.2.10	max_fc_capacity	14
3.8.2.11	metadata_sent	15
3.8.2.12	min_fc_capacity	15
3.8.2.13	out	15
3.8.2.14	posts_mode	15
3.8.2.15	prng	15
3.8.2.16	ra	15
3.8.2.17	read	15
3.8.2.18	remaining	15
3.8.2.19	res_lineup	15
3.8.2.20	res_max_capacity	15
3.8.2.21	res_min_capacity	16
3.8.2.22	res_occupied	16
3.8.2.23	sent	16
3.8.2.24	status	16
3.8.2.25	synchro_key	16
3.8.2.26	synchro_method	16
3.8.2.27	synchronize	16
3.8.2.28	total_sub_capacity	16
3.8.2.29	variation_limit	16
3.8.2.30	vc	16
3.9	vorbis_config_t Struct Reference	17
3.9.1	Detailed Description	17
3.9.2	Member Data Documentation	17
3.9.2.1	forward_index	17
3.9.2.2	mult	17
3.9.2.3	pcmend	17
3.9.2.4	postlist	17
3.9.2.5	posts_len	17
3.9.2.6	rate	17
<b>4</b>	<b>File Documentation</b>	<b>19</b>
4.1	include/cryptos_types.h File Reference	19
4.1.1	Define Documentation	19
4.1.1.1	I_CRYPTOS_CHECK_FAIL	19
4.1.1.2	I_CRYPTOS_ERR	20
4.1.1.3	I_CRYPTOS_OK	20

4.2	include/global_types.h File Reference	21
4.2.1	Define Documentation	21
4.2.1.1	BITS_PER_BYTE	21
4.2.1.2	I_ERR	21
4.2.1.3	I_OK	21
4.2.2	Typedef Documentation	21
4.2.2.1	byte	21
4.3	include/numbers.h File Reference	22
4.3.1	Function Documentation	22
4.3.1.1	linear_interpolation_y	22
4.3.1.2	prng_free	23
4.3.1.3	prng_get_random_int	23
4.3.1.4	prng_init	24
4.3.1.5	prng_reset_byte	24
4.3.1.6	prng_set_seed_byte	24
4.3.1.7	prng_set_seed_int	25
4.3.1.8	seek	25
4.3.1.9	seek_interval	26
4.4	include/steganos_types.h File Reference	27
4.4.1	Define Documentation	28
4.4.1.1	I_STEGANOS_ERR	28
4.4.1.2	I_STEGANOS_FRAME_SKIP	28
4.4.1.3	I_STEGANOS_OK	28
4.4.1.4	I_STEGANOS_SYNC_FAIL	28
4.4.1.5	INDETERMINATE_MARK	28
4.4.1.6	VIF_POSIT	28
4.4.1.7	VORBIS_MAX_BLOCK	28
4.4.2	Typedef Documentation	29
4.4.2.1	desync_method	29
4.4.2.2	hide_method	29
4.4.2.3	sync_method	30
4.4.3	Enumeration Type Documentation	31
4.4.3.1	hide_method_et	31
4.4.3.2	synchro_method_et	31
4.5	lib/cryptos_channel.c File Reference	32
4.6	lib/cryptos_channel.h File Reference	33

4.6.1	Define Documentation	34
4.6.1.1	CRYPTOS_BUFFER_MAX_SIZE	34
4.6.1.2	CRYPTOS_DEFAULT_DATA_SIZE	34
4.6.1.3	CRYPTOS_EMISSION_HEADER_LEN	35
4.6.1.4	CRYPTOS_IV_HEADER_LEN	35
4.6.1.5	CRYPTOS_LENGTH_HEADER_LEN	35
4.6.1.6	CRYPTOS_MAX_DATA	35
4.6.1.7	CRYPTOS_MAX_DIGEST_LEN	35
4.6.1.8	CRYPTOS_MIN_DIGEST_LEN	35
4.6.1.9	CRYPTOS_PACKET_HEADER_LEN	35
4.6.1.10	CRYPTOS_SYNC_HEADER_LEN	35
4.6.1.11	RATIO_DD	35
4.6.2	Function Documentation	35
4.6.2.1	cryptos_buffer_free	35
4.6.2.2	cryptos_buffer_init	36
4.6.2.3	cryptos_cipher_algo_code	36
4.6.2.4	cryptos_config_free	37
4.6.2.5	cryptos_config_init	37
4.6.2.6	cryptos_config_set_iv	38
4.6.2.7	cryptos_config_set_key	38
4.6.2.8	cryptos_key_free	38
4.6.2.9	cryptos_key_init	39
4.6.2.10	cryptos_md_algo_code	39
4.6.2.11	parse_packet	39
4.6.2.12	produce_packet	40
4.7	lib/miscellaneous.c File Reference	42
4.8	lib/miscellaneous.h File Reference	43
4.8.1	Define Documentation	44
4.8.1.1	I_MISC_ERR	44
4.8.1.2	I_MISC_INVALID_XML	44
4.8.1.3	I_MISC_OK	44
4.8.1.4	XML_AGGRESS	44
4.8.1.5	XML_HIDEMETHOD	44
4.8.1.6	XML_KEY	44
4.8.1.7	XML_KEY_BITSTREAM	44
4.8.1.8	XML_KEY_LENGTH	44

4.8.1.9	XML_LEAFS	45
4.8.1.10	XML_SYNCHROMODE	45
4.8.2	Function Documentation	45
4.8.2.1	bitstream_rol	45
4.8.2.2	bitstream_ror	45
4.8.2.3	message_log	46
4.8.2.4	print_bytestream	46
4.8.2.5	to_byte	46
4.8.2.6	zlib_def	47
4.8.2.7	zlib_inf	47
4.8.2.8	zlib_zerr	47
4.9	lib/numbers.c File Reference	48
4.10	lib/protocols.c File Reference	49
4.11	lib/protocols.h File Reference	50
4.11.1	Function Documentation	50
4.11.1.1	cryptos_forward	50
4.11.1.2	cryptos_inverse	51
4.11.1.3	steganos_forward	51
4.11.1.4	steganos_inverse	52
4.11.1.5	steganos_vorbis_config_free	52
4.11.1.6	steganos_vorbis_config_init	53
4.12	lib/steganos_channel.c File Reference	54
4.13	lib/steganos_channel.h File Reference	55
4.13.1	Define Documentation	57
4.13.1.1	BITS_PARITY	57
4.13.1.2	DISTRESS_SYNC_BITS	57
4.13.1.3	DISTRESS_SYNC_BYTES	57
4.13.1.4	FILE_SIZE_BITS	57
4.13.1.5	FILE_SIZE_BYTES	57
4.13.1.6	ITU_R_BS_468_SIZE	57
4.13.1.7	MAX_SUBLIMINAL_SIZE	57
4.13.1.8	RES_HEADER_BITS	57
4.13.1.9	SIZE_FIELD_BITS	58
4.13.1.10	SYNCHRO_HEADER_BYTES_RES	58
4.13.2	Function Documentation	58
4.13.2.1	calculate_residue_lineup	58



4.13.2.2	desynchro_iss	58
4.13.2.3	desynchro_res_header	59
4.13.2.4	hide_data	60
4.13.2.5	iss_cfg_free	60
4.13.2.6	iss_cfg_init	61
4.13.2.7	iss_simulate_floor	61
4.13.2.8	parity_bits_method	62
4.13.2.9	set_subliminal_capacity_limit	62
4.13.2.10	steganos_free_packet_keys	63
4.13.2.11	steganos_key_init	63
4.13.2.12	steganos_prepare_packet_keys	64
4.13.2.13	steganos_state_free	64
4.13.2.14	steganos_state_init	64
4.13.2.15	steganos_state_reset_iter	65
4.13.2.16	synchro_iss	65
4.13.2.17	unhide_data	66
4.13.2.18	void_method	67



# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">cryptos_config_t</a> (Defines the global configuration of the cryptographic protocol ) . . . . .	5
<a href="#">cryptos_key_t</a> (Defines the key structure to be used in the cryptographic layer ) . . . . .	7
<a href="#">cryptos_protocol_buffer</a> (Buffer used within the cryptos protocol and which also serves as inter- face with the steganos protocol ) . . . . .	8
<a href="#">cryptos_protocol_buffer_t</a> . . . . .	9
<a href="#">iss_cfg_t</a> . . . . .	10
<a href="#">prng_t</a> (Defines the structure to use as PRNG abstraction ) . . . . .	11
<a href="#">steganos_key_t</a> (Defines the key structure to be used in the steganographic layer ) . . . . .	12
<a href="#">steganos_state_t</a> (Defines the global internal state of the steganosgraphic protocol ) . . . . .	13
<a href="#">vorbis_config_t</a> (Stores the needed information about the Vorbis block and look configuration. Is used to avoid circular dependencies ) . . . . .	17



# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">include/cryptos_types.h</a>	19
<a href="#">include/global_types.h</a>	21
<a href="#">include/numbers.h</a>	22
<a href="#">include/steganos_types.h</a>	27
<a href="#">lib/cryptos_channel.c</a>	32
<a href="#">lib/cryptos_channel.h</a>	33
<a href="#">lib/miscellaneous.c</a>	42
<a href="#">lib/miscellaneous.h</a>	43
<a href="#">lib/numbers.c</a>	48
<a href="#">lib/protocols.c</a>	49
<a href="#">lib/protocols.h</a>	50
<a href="#">lib/steganos_channel.c</a>	54
<a href="#">lib/steganos_channel.h</a>	55



# Chapter 3

## Class Documentation

### 3.1 cryptos\_config\_t Struct Reference

Defines the global configuration of the cryptographic protocol.

```
#include <include/cryptos_types.h>
```

#### Public Attributes

- int [cipher\\_algo](#)
- int [md\\_algo](#)
- int [md\\_len](#)
- int [max\\_data](#)
- int [hmac](#)
- gcry\_cipher\_hd\_t [chd](#)
- gcry\_md\_hd\_t [mdhd](#)
- [cryptos\\_key\\_t](#) \* [key](#)
- [cryptos\\_key\\_t](#) \* [iv](#)
- uint64\_t [emission](#)
- uint64\_t [packet](#)
- uint64\_t [default\\_data\\_size](#)

#### 3.1.1 Detailed Description

Defines the global configuration of the cryptographic protocol.

See also:

[cryptos\\_key\\_t](#)

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 gcry\_cipher\_hd\_t cryptos\_config\_t::chd

Handler for the ciphering algorithm

**3.1.2.2 int cryptos\_config\_t::cipher\_algo**

Cipher algorithm to use

**3.1.2.3 uint64\_t cryptos\_config\_t::default\_data\_size**

Default size of data, in bytes, to send in a given packet.

**3.1.2.4 uint64\_t cryptos\_config\_t::emission**

Current emission ID

**3.1.2.5 int cryptos\_config\_t::hmac**

When active, indicate hmac variant of digest algorithm

**3.1.2.6 cryptos\_key\_t\* cryptos\_config\_t::iv**

IV to use in the cipher algorithm

**3.1.2.7 cryptos\_key\_t\* cryptos\_config\_t::key****3.1.2.8 int cryptos\_config\_t::max\_data**

Max data to hide. Will depend on the cipher and digest algorithms

**3.1.2.9 int cryptos\_config\_t::md\_algo**

Digest algorithm to use

**3.1.2.10 int cryptos\_config\_t::md\_len**

Digest length

**3.1.2.11 gcry\_md\_hd\_t cryptos\_config\_t::mdhd**

Handler for the message digest algorithm

**3.1.2.12 uint64\_t cryptos\_config\_t::packet**

Current packet ID

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

- [include/cryptos\\_types.h](#)



## 3.2 cryptos\_key\_t Struct Reference

Defines the key structure to be used in the cryptographic layer.

```
#include <include/cryptos_types.h>
```

### Public Attributes

- [byte \\* key](#)
- [int length](#)

### 3.2.1 Detailed Description

Defines the key structure to be used in the cryptographic layer.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 [byte\\* cryptos\\_key\\_t::key](#)

Key value

#### 3.2.2.2 [int cryptos\\_key\\_t::length](#)

Key length, in bytes.

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

- [include/cryptos\\_types.h](#)

## 3.3 cryptos\_protocol\_buffer Struct Reference

Buffer used within the cryptos protocol and which also serves as interface with the steganos protocol.

```
#include <cryptos_types.h>
```

### 3.3.1 Detailed Description

Buffer used within the cryptos protocol and which also serves as interface with the steganos protocol.

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

- [include/cryptos\\_types.h](#)

## 3.4 cryptos\_protocol\_buffer\_t Struct Reference

```
#include <cryptos_types.h>
```

### Public Attributes

- `int` `fd`
- `size_t` `offset`
- `byte *` `buffer`
- `size_t` `buffer_size`
- `size_t` `buffer_used`

### 3.4.1 Member Data Documentation

#### 3.4.1.1 `byte* cryptos_protocol_buffer_t::buffer`

Buffer to store data not successfully sent or received in previous packets.

#### 3.4.1.2 `size_t cryptos_protocol_buffer_t::buffer_size`

Allocated size for buffer.

#### 3.4.1.3 `size_t cryptos_protocol_buffer_t::buffer_used`

Current amount of bytes in buffer.

#### 3.4.1.4 `int cryptos_protocol_buffer_t::fd`

The file from which we'll read the data to send at the emitter's side, or the file in which we'll write the data at the receiver's side.

#### 3.4.1.5 `size_t cryptos_protocol_buffer_t::offset`

Offset inside fd, in bytes.

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

- `include/cryptos_types.h`

## 3.5 iss\_cfg\_t Struct Reference

```
#include <steganos_types.h>
```

### Public Attributes

- float [alpha](#)
- float [lambda](#)
- float [sigma](#)
- float \* [u](#)
- float [u\\_norm](#)

### 3.5.1 Member Data Documentation

#### 3.5.1.1 float iss\_cfg\_t::alpha

$\alpha$  in  $s = x + (\alpha * b - \lambda * x) * u$

#### 3.5.1.2 float iss\_cfg\_t::lambda

$\lambda$  in  $s = x + (\alpha * b - \lambda * x) * u$

#### 3.5.1.3 float iss\_cfg\_t::sigma

The deviation to use in the watermarking random sequence

#### 3.5.1.4 float\* iss\_cfg\_t::u

The watermark

#### 3.5.1.5 float iss\_cfg\_t::u\_norm

Norm of the watermark

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

- [include/steganos\\_types.h](#)

## 3.6 prng\_t Struct Reference

Defines the structure to use as PRNG abstraction.

```
#include <include/numbers.h>
```

### Public Attributes

- void \* [seed](#)
- long int [iters](#)

### 3.6.1 Detailed Description

Defines the structure to use as PRNG abstraction.

### 3.6.2 Member Data Documentation

#### 3.6.2.1 long int prng\_t::iters

Numbers of times used

#### 3.6.2.2 void\* prng\_t::seed

Seed used in the PRNG

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

- [include/numbers.h](#)

## 3.7 steganos\_key\_t Struct Reference

Defines the key structure to be used in the steganographic layer.

```
#include <include/steganos_types.h>
```

### Public Attributes

- [byte \\* key](#)
- [int length](#)

### 3.7.1 Detailed Description

Defines the key structure to be used in the steganographic layer.

### 3.7.2 Member Data Documentation

#### 3.7.2.1 [byte\\* steganos\\_key\\_t::key](#)

Key value, of size  $\lceil length / BITS\_PER\_BYTE \rceil$  bytes.

#### 3.7.2.2 [int steganos\\_key\\_t::length](#)

Key length, in bits.

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

- [include/steganos\\_types.h](#)

## 3.8 steganos\_state\_t Struct Reference

Defines the global internal state of the steganosgraphic protocol.

```
#include <include/steganos_types.h>
```

### Public Attributes

- [vorbis\\_config\\_t](#) \* [vc](#)
- [steganos\\_key\\_t](#) \* [master\\_key](#)
- [steganos\\_key\\_t](#) \* [synchro\\_key](#)
- [steganos\\_key\\_t](#) \* [hiding\\_key](#)
- [synchro\\_method\\_et](#) [synchro\\_method](#)
- [sync\\_method](#) [synchronize](#)
- [desync\\_method](#) [desynchronize](#)
- [int](#) [desync](#)
- [hide\\_method\\_et](#) [hide\\_method](#)
- [hide\\_method](#) [hide](#)
- [unsigned int](#) [da](#)
- [float](#) [ra](#)
- [float](#) [variation\\_limit](#) [VORBIS\_MAX\_BLOCK][2]
- [float](#) [max\\_fc\\_capacity](#)
- [float](#) [min\\_fc\\_capacity](#)
- [int](#) [res\\_max\\_capacity](#) [VORBIS\_MAX\_BLOCK]
- [int](#) [res\\_min\\_capacity](#) [VORBIS\_MAX\_BLOCK]
- [int](#) [res\\_lineup](#) [VORBIS\_MAX\_BLOCK]
- [int](#) [res\\_occupied](#) [VORBIS\_MAX\_BLOCK]
- [unsigned long int](#) [sent](#)
- [unsigned long int](#) [read](#)
- [long int](#) [metadata\\_sent](#)
- [long int](#) [remaining](#)
- [int](#) [out](#) [VIF\_POSIT+2]
- [int](#) [status](#)
- [int](#) [posts\\_mode](#)
- [int](#) [aligned](#)
- [long int](#) [total\\_sub\\_capacity](#)
- [int](#) [iters](#)
- [prng\\_t](#) \* [prng](#)

### 3.8.1 Detailed Description

Defines the global internal state of the steganosgraphic protocol.

See also:

[synchro\\_method\\_et](#)  
[sync\\_method](#)  
[desync\\_method](#)  
[hide\\_method\\_et](#)  
[hide\\_method](#)  
[steganos\\_key\\_t](#)

## 3.8.2 Member Data Documentation

### 3.8.2.1 `int steganos_state_t::aligned`

Boolean signaling that we've already calculated the residue lineup. Avoids recalculating it when considering several different stego-frame-channels.

### 3.8.2.2 `unsigned int steganos_state_t::da`

Desired aggressiveness of the protocol. Remains the same through the whole process. Ranges from 1 to 10, meaning we want to use the 10,20, ... 100% of the total subliminal capacity.

### 3.8.2.3 `int steganos_state_t::desync`

Boolean. Active when desynchronization must be done.

### 3.8.2.4 `desync_method steganos_state_t::desynchronize`

Desynchronization method (pointer to function)

### 3.8.2.5 `hide_method steganos_state_t::hide`

Hiding method (pointer to function) to use in the current frame and channel.

### 3.8.2.6 `hide_method_et steganos_state_t::hide_method`

Hiding method (enum type) to use in the current frame and channel residue vectors.

### 3.8.2.7 `steganos_key_t* steganos_state_t::hiding_key`

Key to use for hiding the data in the residues.

### 3.8.2.8 `int steganos_state_t::iters`

Usefull when debugging and printing statistics, stores how many iterations of the steganographic protocol has been run at a given instant.

### 3.8.2.9 `steganos_key_t* steganos_state_t::master_key`

Master key used to derive the subkeys.

### 3.8.2.10 `float steganos_state_t::max_fc_capacity`

Stores the maximum subliminal capacity for the current frame and channel residue, in bits.



**3.8.2.11 long int steganos\_state\_t::metadata\_sent**

The same as sent, but including metadata

**3.8.2.12 float steganos\_state\_t::min\_fc\_capacity**

Stores the minimum subliminal capacity for the current frame and channel residue, in bits.

**3.8.2.13 int steganos\_state\_t::out[VIF\_POSIT+2]**

Vector with the posts values to write in the ogg packet when using ISS.

**3.8.2.14 int steganos\_state\_t::posts\_mode**

With ISS, this variable indicates the way we're trying to mark the floor posts. 1 means we're trying to mark it with a 1 bit, 0 means we're using the original posts, and -1 means we're trying to mark it with a 0 bit.

**3.8.2.15 prng\_t\* steganos\_state\_t::prng**

PRNG abstraction

**3.8.2.16 float steganos\_state\_t::ra**

The aggressiveness is used as a guide to the global share of the subliminal channel we desire to use, but due to the stochastic nature of the method, we shall use it as a guide rather than a strict quantity. The *real aggressiveness* will change from channel to channel and will be used as a correction indicator.

**3.8.2.17 unsigned long int steganos\_state\_t::read**

Total amount of \_pure\_ data (excluding metadata) already recovered from the stego-audio stream.

**3.8.2.18 long int steganos\_state\_t::remaining**

Total amount of \_pure\_ data to send/receive.

**3.8.2.19 int steganos\_state\_t::res\_lineup[VORBIS\_MAX\_BLOCK]**

Will store the ordering of the residual elements for hiding/unhiding purposes.

**3.8.2.20 int steganos\_state\_t::res\_max\_capacity[VORBIS\_MAX\_BLOCK]**

Will store the maximum number of subliminal bits a given residual element should shelter. This value should be seen also as a guide.

**3.8.2.21 int steganos\_state\_t::res\_min\_capacity[VORBIS\_MAX\_BLOCK]**

Will store the minimum number of subliminal bits a given residual element should shelter. This value should be seen also as a guide.

**3.8.2.22 int steganos\_state\_t::res\_occupied[VORBIS\_MAX\_BLOCK]**

Will be used to mark the residual values which will be used to write in or read from within the same frame. This array will prevent from writing/ reading in/from the same residual value more than once.

**3.8.2.23 unsigned long int steganos\_state\_t::sent**

Total amount of `_pure_data` (excluding metadata) already hided in the audio stream.

**3.8.2.24 int steganos\_state\_t::status**

Status code. Used to control "inter fuction" errors

**3.8.2.25 steganos\_key\_t\* steganos\_state\_t::synchro\_key**

Key to use for synchronization.

**3.8.2.26 synchro\_method\_et steganos\_state\_t::synchro\_method**

Synchronization method (enum type). Can change from one frame and channel to another.

**3.8.2.27 sync\_method steganos\_state\_t::synchronize**

Synchronization method (pointer to function) to use in the current frame and channel.

**3.8.2.28 long int steganos\_state\_t::total\_sub\_capacity**

The total subliminal capacity at a given instant. Useful for statistics and debugging. Only meaningful when encoding.

**3.8.2.29 float steganos\_state\_t::variation\_limit[VORBIS\_MAX\_BLOCK][2]**

Variation limit per residual line. Stores the `[-, +]` maximum variations to introduce in each residual value. Changes from one frame and channel to another.

**3.8.2.30 vorbis\_config\_t\* steganos\_state\_t::vc**

Vorbis configuration data needed during the steganographic protocol.

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

- [include/steganos\\_types.h](#)

## 3.9 vorbis\_config\_t Struct Reference

Stores the needed information about the Vorbis block and look configuration. Is used to avoid circular dependencies.

```
#include <global_types.h>
```

### Public Attributes

- int [rate](#)
- int [pcmend](#)
- int [mult](#)
- int \* [postlist](#)
- int \* [forward\\_index](#)
- int [posts\\_len](#)

### 3.9.1 Detailed Description

Stores the needed information about the Vorbis block and look configuration. Is used to avoid circular dependencies.

### 3.9.2 Member Data Documentation

**3.9.2.1** int\* [vorbis\\_config\\_t::forward\\_index](#)

**3.9.2.2** int [vorbis\\_config\\_t::mult](#)

**3.9.2.3** int [vorbis\\_config\\_t::pcmend](#)

**3.9.2.4** int\* [vorbis\\_config\\_t::postlist](#)

**3.9.2.5** int [vorbis\\_config\\_t::posts\\_len](#)

**3.9.2.6** int [vorbis\\_config\\_t::rate](#)

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

- [include/global\\_types.h](#)



# Chapter 4

## File Documentation

### 4.1 include/cryptos\_types.h File Reference

```
#include <gcrypt.h>
#include "global_types.h"
```

#### Classes

- struct [cryptos\\_key\\_t](#)  
*Defines the key structure to be used in the cryptographic layer.*
- struct [cryptos\\_protocol\\_buffer\\_t](#)
- struct [cryptos\\_config\\_t](#)  
*Defines the global configuration of the cryptographic protocol.*

#### Defines

- #define [I\\_CRYPTOS\\_OK](#) 0  
*Error code for 'No error' to use in functions of integer return type.*
- #define [I\\_CRYPTOS\\_ERR](#) 1  
*Error code for 'Error occurred' to use in functions of integer return type.*
- #define [I\\_CRYPTOS\\_CHECK\\_FAIL](#) 2  
*Error code for 'Integrity check failed'.*

#### 4.1.1 Define Documentation

##### 4.1.1.1 #define I\_CRYPTOS\_CHECK\_FAIL 2

Error code for 'Integrity check failed'.

**4.1.1.2 #define I\_CRYPTOS\_ERR 1**

Error code for 'Error occurred' to use in functions of integer return type.

**4.1.1.3 #define I\_CRYPTOS\_OK 0**

Error code for 'No error' to use in functions of integer return type.

## 4.2 include/global\_types.h File Reference

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

### Classes

- struct [vorbis\\_config\\_t](#)

*Stores the needed information about the Vorbis block and look configuration. Is used to avoid circular dependencies.*

### Defines

- #define [I\\_OK](#) 0  
*Error code for 'No error' to use in functions of integer return type.*
- #define [I\\_ERR](#) 1  
*Error code for 'Error occurred' to use in functions of integer return type.*
- #define [BITS\\_PER\\_BYTE](#) 8  
*Modify for different byte sizes.*

### Typedefs

- typedef unsigned char [byte](#)

#### 4.2.1 Define Documentation

##### 4.2.1.1 #define [BITS\\_PER\\_BYTE](#) 8

Modify for different byte sizes.

##### 4.2.1.2 #define [I\\_ERR](#) 1

Error code for 'Error occurred' to use in functions of integer return type.

##### 4.2.1.3 #define [I\\_OK](#) 0

Error code for 'No error' to use in functions of integer return type.

#### 4.2.2 Typedef Documentation

##### 4.2.2.1 typedef unsigned char [byte](#)

## 4.3 include/numbers.h File Reference

```
#include "global_types.h"
```

### Classes

- struct [prng\\_t](#)  
*Defines the structure to use as PRNG abstraction.*

### Functions

- int [prng\\_init](#) ([prng\\_t](#) \*prng)  
*Initializes the PRNG.*
- int [prng\\_free](#) ([prng\\_t](#) \*prng)  
*Sets the seed to use to the seed, of size bits.*
- int [prng\\_reset\\_byte](#) ([prng\\_t](#) \*prng, [byte](#) \*seed, int size, long int iters)  
*Recalculate the state of the given PRNG seeded with seed and used prng->iters times.*
- int [prng\\_set\\_seed\\_byte](#) ([prng\\_t](#) \*prng, const [byte](#) \*seed, const int size)  
*Sets the seed to use to the seed, of size bits.*
- int [prng\\_set\\_seed\\_int](#) ([prng\\_t](#) \*prng, const int seed)  
*Sets the seed to use to the seed.*
- int [prng\\_get\\_random\\_int](#) ([prng\\_t](#) \*prng, const int modulo, int \*r)  
*Feeds r with a pseudo random integer.*
- int [linear\\_interpolation\\_y](#) (const float x1, const float y1, const float x2, const float y2, const float x, float \*y)  
*Stores the result of the lineal interpolation between the points [x1,y1] and [x2,y2] in [x,y].*
- int [seek](#) (int \*sequence, int seq\_len, int number, int \*exist)  
*Looks for number in sequence, updating exist in consequence.*
- int [seek\\_interval](#) (int \*sequence, int seq\_len, int lower, int higher, int \*exist)  
*Looks if there is any number in sequence belonging to the interval defined by [lower, higher], updating exist in consequence.*

### 4.3.1 Function Documentation

#### 4.3.1.1 int [linear\\_interpolation\\_y](#) (const float x1, const float y1, const float x2, const float y2, const float x, float \*y)

Stores the result of the lineal interpolation between the points [x1,y1] and [x2,y2] in [x,y]. Given [x1, y1], [x2, y2] and x, stores the result of the lineal interpolation between the points [x1,y1] and [x2,y2] in [x,y].



**Parameters:**

- ← *x1* The first point x-coordinate.
- ← *y1* The first point y-coordinate.
- ← *x2* The second point x-coordinate.
- ← *y2* The second point y-coordinate.
- ← *x* The x coordinate of the new point.
- *y* Will store the resulting new point y-coordinate.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_OK` if no error was present and `I_ERR` if an error occurred with `errno` updated.

**Return values:**

- `I_OK` with `errno = 0` (No error).
- `I_ERR` with `errno = EINVAL` (invalid argument).

**4.3.1.2 int prng\_free (prng\_t \* prng)**

Sets the seed to use to the *seed*, of *size* bits. Frees the memory allocated for the given PRNG

**Parameters:**

- ← *prng* PRNG to free.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_OK` if no error was present and `I_ERR` if an error occurred with `errno` updated.

**Return values:**

- `I_OK` with `errno = 0` (No error).

**4.3.1.3 int prng\_get\_random\_int (prng\_t \* prng, const int modulo, int \* r)**

Feeds *r* with a pseudo random integer. Obtains a new pseudo random number and stores it in *r*.

**Parameters:**

- ← *prng* PRNG abstraction pointer.
- ← *modulo* The pseudo random number generated will take a value between 0 and *modulo*-1. If *modulo* equals 0, the function won't apply any modular arithmetic.
- *r* Will store the pseudo random number generated.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_OK` if no error was present and `I_ERR` if an error occurred with `errno` updated.

**Return values:**

- `I_OK` with `errno = 0` (No error).
- `I_ERR` with `errno = EINVAL` (invalid argument).

#### 4.3.1.4 int prng\_init (prng\_t \* *prng*)

Initializes the PRNG. Allocates for the given PRNG

##### Parameters:

← *prng* PRNG to allocate.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_OK if no error was present and I\_ERR if an error occurred with errno updated.

##### Return values:

**I\_OK** with errno = 0 (No error).

**I\_ERR** with errno = EINVAL (invalid argument).

#### 4.3.1.5 int prng\_reset\_byte (prng\_t \* *prng*, byte \* *seed*, int *size*, long int *iters*)

Recalculate the state of the given PRNG seeded with *seed* and used *prng*->*iters* times. Recalculates the state of the PRNG *prng* after being seeded with *seed* and iterated *iters* times.

##### Parameters:

← *prng* The PRNG.

← *seed* The seed used to initialize prng.

← *size* The size of seed, in bits.

← *iters* The iterations to "discard".

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_OK if no error was present and I\_ERR if an error occurred with errno updated.

##### Return values:

**I\_OK** with errno = 0 (No error).

**I\_ERR** with errno = EINVAL (invalid argument).

#### 4.3.1.6 int prng\_set\_seed\_byte (prng\_t \* *prng*, const byte \* *seed*, const int *size*)

Sets the seed to use to the *seed*, of *size* bits. Sets the seed to use for generating [CS]PRN sequences to *seed*, which will have *size* bits of length.

##### Parameters:

← *prng* PRNG abstraction pointer.

← *seed* The seed to use, in a byte representation

← *size* The size of the seed, in bits

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_OK` if no error was present and `I_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_OK*** with `errno = 0` (No error).

***I\_ERR*** with `errno = EINVAL` (invalid argument).

**4.3.1.7 int prng\_set\_seed\_int (prng\_t \* prng, const int seed)**

Sets the seed to use to the *seed*. Sets the seed to use for generating [CS]PRN sequences to *seed*, which will have *size* bits of length.

**Parameters:**

← ***prng*** PRNG abstraction pointer.

← ***seed*** The seed to use, in an int representation

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_OK` if no error was present and `I_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_OK*** with `errno = 0` (No error).

***I\_ERR*** with `errno = EINVAL` (invalid argument).

**4.3.1.8 int seek (int \* sequence, int seq\_len, int number, int \* exist)**

Looks for *number* in *sequence*, updating *exist* in consequence. Looks for *number* in *sequence*. After a successful run, *exist* will equal to 1 if the number exists and 0 otherwise.

**Parameters:**

← ***sequence*** A list of numbers.

← ***seq\_len*** The number of elements in sequence.

← ***number*** The number to look for.

→ ***exist*** Will be 1 if the number exist and 0 otherwise, in a successful run of the function.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_OK` if no error was present and `I_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_OK*** with `errno = 0` (No error).

***I\_ERR*** with `errno = EINVAL` (invalid argument).

**4.3.1.9 int seek\_interval (int \* *sequence*, int *seq\_len*, int *lower*, int *higher*, int \* *exist*)**

Looks if there is any number in *sequence* belonging to the interval defined by [*lower*; *higher*], updating *exist* in consequence. Looks if there is any number in *sequence* belonging to the interval defined by [*lower*; *higher*]. After a successful run, *exist* will equal to 1 if the number exists and 0 otherwise.

**Parameters:**

- ← *sequence* A list of numbers.
- ← *seq\_len* The number of elements in sequence.
- ← *lower* The lower end of the interval.
- ← *higher* The higher end of the interval.
- *exist* Will be 1 if the number exist and 0 otherwise, in a successful run of the function.

**Returns:**

The corresponding error code for integer returning functions, i.e., I\_OK if no error was present and I\_ERR if an error occurred with *errno* updated.

**Return values:**

- I\_OK* with *errno* = 0 (No error).
- I\_ERR* with *errno* = EINVAL (invalid argument).

## 4.4 include/steganos\_types.h File Reference

```
#include "global_types.h"
#include "numbers.h"
```

### Classes

- struct [steganos\\_key\\_t](#)  
*Defines the key structure to be used in the steganographic layer.*
- struct [iss\\_cfg\\_t](#)
- struct [steganos\\_state\\_t](#)  
*Defines the global internal state of the steganographic protocol.*

### Defines

- #define [L\\_STEGANOS\\_OK](#) L\_OK  
*Error code for 'No error' to use in functions of integer return type.*
- #define [L\\_STEGANOS\\_ERR](#) L\_ERR  
*Error code for 'Error occurred' to use in functions of integer return type.*
- #define [L\\_STEGANOS\\_SYNC\\_FAIL](#) 2  
*Error code for 'Unable to synchronize', used when a synchronization method is unable to synchronize due to its own nature, i.e., there is no computation error.*
- #define [L\\_STEGANOS\\_FRAME\\_SKIP](#) 3  
*Return code used to specify we're not using the current frame to hide data. This can be due, for example, to a short subliminal capacity of the frame or to "imperceptibility" reasons.*
- #define [INDETERMINATE\\_MARK](#) -1  
*Value used when there is not enough statistical evidence of the mark hidden in the Vorbis's posts vector being 0 or 1. Used only with ISS synchronization method.*
- #define [VORBIS\\_MAX\\_BLOCK](#) 8192  
*Defines the Vorbis maximum number of samples per Vorbis block.*
- #define [VIF\\_POSIT](#) 63

### Typedefs

- typedef int(\* [hide\\_method](#) )(byte \*plain\_mess, int p\_m\_len, int \*floor, float \*res, int res\_len, [steganos\\_key\\_t](#) \*key, byte \*sub\_mess, int \*s\_m\_len, [prng\\_t](#) \*prng)  
*Type definition for hiding functions to use in the residual subliminal channel.*
- typedef int(\* [sync\\_method](#) )(vorbis\_config\_t \*vc, [steganos\\_key\\_t](#) \*key, int \*posts, float \*residue, void \*cfg, int decoding, int \*bit, [prng\\_t](#) \*prng)  
*Type definition for synchronization functions to use in the protocol.*

- typedef int(\* [desync\\_method](#))([vorbis\\_config\\_t](#) \*vc, int \*res\_lineup, int \*posts, int \*floor, float \*residue, void \*cfg, [steganos\\_key\\_t](#) \*hiding\_key, [hide\\_method](#) hide, [prng\\_t](#) \*prng)

*Type definition for desynchronization functions to use in the protocol.*

## Enumerations

- enum [synchro\\_method\\_et](#) { [RES\\_HEADER](#) = 0, [ISS](#) = 1, [FORCED\\_RES\\_HEADER](#) = 2 }

*Enumeration for the different synchronization modes available.*

- enum [hide\\_method\\_et](#) { [DIRECT\\_HIDING](#) = 0, [PARITY\\_BITS](#) = 1 }

*Enumeration for the different methods of hiding data in the residues.*

### 4.4.1 Define Documentation

#### 4.4.1.1 #define I\_STEGANOS\_ERR I\_ERR

Error code for 'Error occurred' to use in functions of integer return type.

#### 4.4.1.2 #define I\_STEGANOS\_FRAME\_SKIP 3

Return code used to specify we're not using the current frame to hide data. This can be due, for example, to a short subliminal capacity of the frame or to "imperceptibility" reasons.

#### 4.4.1.3 #define I\_STEGANOS\_OK I\_OK

Error code for 'No error' to use in functions of integer return type.

#### 4.4.1.4 #define I\_STEGANOS\_SYNC\_FAIL 2

Error code for 'Unable to synchronize', used when a synchronization method is unable to synchronize due to its own nature, i.e., there is no computation error.

#### 4.4.1.5 #define INDETERMINATE\_MARK -1

Value used when there is not enough statistical evidence of the mark hidden in the Vorbis's posts vector being 0 or 1. Used only with ISS synchronization method.

#### 4.4.1.6 #define VIF\_POSIT 63

#### 4.4.1.7 #define VORBIS\_MAX\_BLOCK 8192

Defines the Vorbis maximum number of samples per Vorbis block.

## 4.4.2 Typedef Documentation

### 4.4.2.1 `int(* desync_method)(vorbis_config_t *vc, int *res_lineup, int *posts, int *floor, float *residue, void *cfg, steganos_key_t *hiding_key, hide_method hide, prng_t *prng)`

Type definition for desynchronization functions to use in the protocol. Functions of this type shall solve the false positive synchronization problem that may happen depending on the `synchro_method` used. Depending on the specific method in use, the *floor* and *residue* vectors will or won't be used.

#### Parameters:

- ← *vorbis\_config\_t* Vorbis configuration data.
- ← *res\_lineup* Residue ordering
- ↔ *posts* Current floor posts vector. May be subject to changes depending on the concrete method used. See synchronization methods.
- ↔ *floor* Current floor vector. May be subject to changes depending on the concrete method used. See synchronization methods.
- ↔ *residue* Current residue vector. May be subject to changes depending on the concrete method used. See synchronization methods.
- ← *cfg* This parameter will be used to pass any argument needed in a concrete synchronization method.
- ← *hiding\_key* Key for hiding
- ← *hide* Hiding method (pointer to function) in use in the current frame and channel.
- ← *prng* PRNG in use

#### Returns:

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

#### Return values:

- I\_STEGANOS\_OK* with `errno = 0` (No error).
- I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).

#### See also:

[synchro\\_method\\_et](#)  
[iss\\_synchronization](#)

### 4.4.2.2 `int(* hide_method)(byte *plain_mess, int p_m_len, int *floor, float *res, int res_len, steganos_key_t *key, byte *sub_mess, int *s_m_len, prng_t *prng)`

Type definition for hiding functions to use in the residual subliminal channel. Functions of this type shall produce a stego message of the same length as the subliminal message, and may or may not use the cover object and/or the key during the process. In case the cover object and/or the key aren't necessary, the corresponding byte \* parameters representing them must be `NULL` and their corresponding int's be 0.

#### Parameters:

- ← *plain\_mess* Plain message to hide.
- ← *p\_m\_len* Length of plain message in bits.

- ← *floor* Floor vector.
- ← *floor\_len* The length of the floor vector (actually, the floor posts in Vorbis).
- ← *res* Residue vector.
- ← *res\_len* Number of elements in *floor* and *res* vectors.
- ← *key* Key to use.
- ↔ *sub\_mess* String in which the subliminal message will be stored. It has to be allocated previously to calling the function.
- ↔ *s\_m\_len* At the input, will store the allocated size (in bits) of *sub\_mess*. At the output, the length of the resulting sub\_message, in bits (it may be different than *p\_m\_len* as it may include de-synchronization bits).
- ← *prng* PRNG in use.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

- I\_STEGANOS\_OK* with `errno = 0` (No error).
- I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).

#### 4.4.2.3 `int(* sync_method)(vorbis_config_t *vc, steganos_key_t *key, int *posts, float *residue, void *cfg, int decoding, int *bit, prng_t *prng)`

Type definition for synchronization functions to use in the protocol. Functions of this type shall solve the synchronization problem between sender and receiver using the "both-known" data (stored in *ss* and the floor and residue vectors, both of length *res\_len*). Depending on the specific method use, the *floor* and *residue* vectors will or won't be used.

**Parameters:**

- ← *vc* Vorbis block and look information.
- ← *key* Key to use
- ↔ *posts* Current floor posts vector. May be subject to changes depending on the concrete method used. See synchronization methods.
- ↔ *residue* Current residue vector. May be subject to changes depending on the concrete method used. See synchronization methods.
- ← *cfg* This parameter will be used to pass any argument needed in a concrete synchronization method.
- ← *decoding* Active if the function is called from the decoder's side,
- ↔ *bit* At the input, the desired bit to include if we are at the encoder's side; at the output, the received bit if we are at the decoder's side (i.e. 1 if we're telling the receiver there is hidden data and 0 if not).
- ← *prng* PRNG in use

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.



**Return values:**

*I\_STEGANOS\_OK* with errno = 0 (No error).  
*I\_STEGANOS\_ERR* with errno = EINVAL (invalid argument).  
*I\_STEGANOS\_SYNC\_FAIL* with errno = 0 (Unable to synchronize).

**See also:**

[synchro\\_method\\_et](#)  
iss\_synchronization

### 4.4.3 Enumeration Type Documentation

#### 4.4.3.1 enum hide\_method\_et

Enumeration for the different methods of hiding data in the residues.

**Enumerator:**

*DIRECT\_HIDING* Just write the data in the given locations.  
*PARITY\_BITS* Applies the parity method by Anderson and Petitcolas.

#### 4.4.3.2 enum synchro\_method\_et

Enumeration for the different synchronization modes available.

**Enumerator:**

*RES\_HEADER* Classic mode: synchronization header in residue subliminal channel  
*ISS* Improved Spread Spectrum algorithm within floor vector (uses synchro\_key)  
*FORCED\_RES\_HEADER* Used when ISS is the default option but for some reason we can't use it at a given moment, and also to prevent false negatives at the decoder's side.

## 4.5 lib/cryptos\_channel.c File Reference

## 4.6 lib/cryptos\_channel.h File Reference

```
#include "cryptos_types.h"
```

### Defines

- `#define CRYPTOS_SYNC_HEADER_LEN 3`  
*Defines the number of bytes of the SYNC header field.*
- `#define CRYPTOS_LENGTH_HEADER_LEN 4`  
*Defines the number of bytes of the "length" header field.*
- `#define CRYPTOS_IV_HEADER_LEN 16`  
*Defines the number of bytes of the IV header field.*
- `#define CRYPTOS_EMISSION_HEADER_LEN sizeof(uint64_t)`  
*Defines the number of bytes of the EMISSION header field.*
- `#define CRYPTOS_PACKET_HEADER_LEN sizeof(uint64_t)`  
*Defines the number of bytes of the PACKET header field.*
- `#define CRYPTOS_MAX_DATA 0xFFFFFFFF`  
*Defines the maximum number of bytes of the data field, which will depend on the CRYPTOS\_LENGTH\_HEADER\_LEN bytes.*
- `#define CRYPTOS_DEFAULT_DATA_SIZE 512`  
*Defines the default size of the data field in crypto packets.*
- `#define CRYPTOS_MIN_DIGEST_LEN 3`  
*Defines the size of the message digest algorithm which produces the minimum length digest, in bytes. Currently, CRC24.*
- `#define CRYPTOS_MAX_DIGEST_LEN 64`  
*Defines the size of the message digest algorithm which produces the maximum length digest, in bytes. Currently, SHA512.*
- `#define RATIO_DD 16`  
*Defines the number of bytes of data per byte of message digest.*
- `#define CRYPTOS_BUFFER_MAX_SIZE CRYPTOS_PACKET_MAX_LEN*2`  
*The max size, in bytes, of the cryptos buffer. It is set to twice the size of a cryptos packet because it'll never store two whole packets.*

### Functions

- `int cryptos_buffer_init (cryptos_protocol_buffer_t *cb, int fd, size_t size)`  
*Initializes cryptographic layer buffer.*

- int `cryptos_buffer_free` (`cryptos_protocol_buffer_t` \*cb)  
*Frees the resources allocated for the cryptographic layer buffer.*
- int `cryptos_config_init` (`cryptos_config_t` \*cc, int cipher\_algo, byte \*key, int keylen, int md\_algo, int hmac, byte \*iv, int ivlen, uint64\_t emission, uint64\_t packet, uint64\_t default\_data\_size)  
*Initializes the internal variables of the `cryptos_config_t` structure needed during the protocol.*
- int `cryptos_config_free` (`cryptos_config_t` \*cc)  
*Frees any internal variable of the structure.*
- int `cryptos_config_set_key` (`cryptos_config_t` \*cc, `cryptos_key_t` \*key)  
*Sets the key attribute of `cryptos_config_t` to the key specified, freeing the previous iv when necessary.*
- int `cryptos_config_set_iv` (`cryptos_config_t` \*cc, `cryptos_key_t` \*iv)  
*Sets the key attribute of `cryptos_config_t` to the key specified, freeing the previous iv when necessary.*
- int `produce_packet` (`cryptos_config_t` \*cc, byte \*data, uint64\_t d\_len, byte \*packet, uint64\_t p\_len, uint64\_t \*w\_data)  
*Produces a new packet ready to be sent.*
- int `parse_packet` (`cryptos_config_t` \*cc, byte \*packet, uint64\_t p\_len, byte \*data, uint64\_t \*d\_len, uint64\_t \*r\_data)  
*Parses the received data, returning the data in it.*
- int `cryptos_key_init` (byte \*byte\_key, const int size, `cryptos_key_t` \*key)  
*Allocates memory for the key fields and sets them to the received values.*
- int `cryptos_key_free` (`cryptos_key_t` \*key)  
*Frees the memory allocated for the internal variables of the given `cryptos_key`.*
- int `cryptos_cipher_algo_code` (const char \*cipher\_algo\_name, int \*code)  
*Returns the GCRY code for the algorithm with `cipher_algo_name` name. If the specified name is NULL, returns the code for ARCFOUR, which is the default cipher algorithm in this system.*
- int `cryptos_md_algo_code` (const char \*md\_algo\_name, int \*code)  
*Returns the GCRY code for the algorithm with `md_algo_name` name. If the specified name is NULL, returns the code for SHA1, which is the default message digest algorithm in this system.*

## 4.6.1 Define Documentation

### 4.6.1.1 #define CRYPTOS\_BUFFER\_MAX\_SIZE CRYPTOS\_PACKET\_MAX\_LEN\*2

The max size, in bytes, of the cryptos buffer. It is set to twice the size of a cryptos packet because it'll never store two whole packets.

### 4.6.1.2 #define CRYPTOS\_DEFAULT\_DATA\_SIZE 512

Defines the default size of the data field in crypto packets.

**4.6.1.3 #define CRYPTOS\_EMISSION\_HEADER\_LEN sizeof(uint64\_t)**

Defines the number of bytes of the EMISSION header field.

**4.6.1.4 #define CRYPTOS\_IV\_HEADER\_LEN 16**

Defines the number of bytes of the IV header field.

**4.6.1.5 #define CRYPTOS\_LENGTH\_HEADER\_LEN 4**

Defines the number of bytes of the "length" header field.

**4.6.1.6 #define CRYPTOS\_MAX\_DATA 0xFFFFFFFF**

Defines the maximum number of bytes of the data field, which will depend on the CRYPTOS\_LENGTH\_HEADER\_LEN bytes.

**4.6.1.7 #define CRYPTOS\_MAX\_DIGEST\_LEN 64**

Defines the size of the message digest algorithm which produces the maximum length digest, in bytes. Currently, SHA512.

**4.6.1.8 #define CRYPTOS\_MIN\_DIGEST\_LEN 3**

Defines the size of the message digest algorithm which produces the minimum length digest, in bytes. Currently, CRC24.

**4.6.1.9 #define CRYPTOS\_PACKET\_HEADER\_LEN sizeof(uint64\_t)**

Defines the number of bytes of the PACKET header field.

**4.6.1.10 #define CRYPTOS\_SYNC\_HEADER\_LEN 3**

Defines the number of bytes of the SYNC header field.

**4.6.1.11 #define RATIO\_DD 16**

Defines the number of bytes of data per byte of message digest.

**4.6.2 Function Documentation****4.6.2.1 int cryptos\_buffer\_free (cryptos\_protocol\_buffer\_t \* cb)**

Frees the resources allocated for the cryptographic layer buffer.

**Parameters:**

← *cb* The cryptographic layer buffer structure to free

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_CRYPTOS\_OK*** with `errno = 0` (No error).

***I\_CRYPTOS\_ERR*** with `errno = EINVAL` (invalid argument).

***I\_CRYPTOS\_ERR*** with `errno != 0 && errno != EINVAL` (see corresponding error code)

**4.6.2.2 int cryptos\_buffer\_init (cryptos\_protocol\_buffer\_t \* cb, int fd, size\_t size)**

Initializes cryptographic layer buffer. Initializes the internal variables of the `cryptos_config_t` structure needed during the protocol.

**Parameters:**

← ***cb*** The cryptographic layer buffer structure to initialize

← ***fd*** The file descriptor from which the cryptographic layer will retrieve the data to produce crypto packets

← ***size*** The desired size of the buffer, in bytes.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_CRYPTOS\_OK*** with `errno = 0` (No error).

***I\_CRYPTOS\_ERR*** with `errno = EINVAL` (invalid argument).

***I\_CRYPTOS\_ERR*** with `errno != 0 && errno != EINVAL` (see corresponding error code)

**4.6.2.3 int cryptos\_cipher\_algo\_code (const char \* cipher\_algo\_name, int \* code)**

Returns the GCRY code for the algorithm with `cipher_algo_name` name. If the specified name is NULL, returns the code for ARCFOUR, which is the default cipher algorithm in this system.

**Parameters:**

← ***cipher\_algo\_name*** The name of the ciphering algorithm, in char\* representation.

→ ***code*** Variable to store the corresponding code.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_CRYPTOS\_OK*** with `errno = 0` (No error).

**4.6.2.4 int cryptos\_config\_free (cryptos\_config\_t \* cc)**

Frees any internal variable of the structure.

**Parameters:**

*cc* The cryptos config structure to free.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

`I_CRYPTOS_OK` with `errno = 0` (No error).

`I_CRYPTOS_ERR` with `errno = EINVAL` (invalid argument).

**4.6.2.5 int cryptos\_config\_init (cryptos\_config\_t \* cc, int cipher\_algo, byte \* key, int keylen, int md\_algo, int hmac, byte \* iv, int ivlen, uint64\_t emission, uint64\_t packet, uint64\_t default\_data\_size)**

Initializes the internal variables of the `cryptos_config_t` structure needed during the protocol. Initializes the internal variables of the `cryptos_config_t` structure needed during the protocol.

**Parameters:**

- ← *cc* The `cryptos_config_t` structure to initialize
- ← *cipher\_algo* The cipher algorithm to use
- ← *key* The key to use, must have at least 16 bytes
- ← *keylen* Length of key, in bytes
- ← *md\_algo* The message digest algorithm
- ← *hmac* Boolean. When set, indicates the hmac variant of the *md\_algo* wants to be used (if possible)
- ← *iv* Initialization vector. It is not a key, but uses the same structure. Must have a length of 16 bytes (128 bits). Will be NULL at the receiver's side.
- ← *ivlen* Length of iv, in bytes. Will be 0 at the receiver's side.
- ← *emission* Emission id
- ← *packet* Packet id initial value. Will be 1 at the receiver's side.
- ← *default\_data\_size* Default data size, in bytes, to send in each packet.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

`I_CRYPTOS_OK` with `errno = 0` (No error).

`I_CRYPTOS_ERR` with `errno = EINVAL` (invalid argument).

`I_CRYPTOS_ERR` with `errno != 0 && errno != EINVAL` (see corresponding error code)

**See also:**

`cryptos_config_t`

#### 4.6.2.6 `int cryptos_config_set_iv (cryptos_config_t * cc, cryptos_key_t * iv)`

Sets the key attribute of `cryptos_config_t` to the key specified, freeing the previous iv when necessary.

**Parameters:**

*cc* The cryptos config structure

*iv* The IV

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

*I\_CRYPTOS\_OK* with `errno` = 0 (No error).

*I\_CRYPTOS\_ERR* with `errno` = `EINVAL` (invalid argument).

#### 4.6.2.7 `int cryptos_config_set_key (cryptos_config_t * cc, cryptos_key_t * key)`

Sets the key attribute of `cryptos_config_t` to the key specified, freeing the previous iv when necessary.

**Parameters:**

*cc* The cryptos config structure

*key* The key

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

*I\_CRYPTOS\_OK* with `errno` = 0 (No error).

*I\_CRYPTOS\_ERR* with `errno` = `EINVAL` (invalid argument).

#### 4.6.2.8 `int cryptos_key_free (cryptos_key_t * key)`

Frees the memory allocated for the internal variables of the given `cryptos_key`.

**Parameters:**

← *key* A pointer to the key to free.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

**Return values:**

*I\_CRYPTOS\_OK* with `errno` = 0 (No error).



#### 4.6.2.9 int cryptos\_key\_init (byte \* *byte\_key*, const int *size*, cryptos\_key\_t \* *key*)

Allocates memory for the key fields and sets them to the received values. Allocates memory for the key fields and sets them to the received values.

##### Parameters:

- ← *byte\_key* The key value.
- ← *size* The size of the *byte\_key* argument, in bytes.
- ← *key* A pointer to a previously allocated `cryptos_key_t` variable in which we'll store the key's members.

##### Returns:

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

##### Return values:

- `I_CRYPTOS_OK` with `errno = 0` (No error).
- `I_CRYPTOS_ERR` with `errno = EINVAL` (invalid argument).
- `I_CRYPTOS_ERR` with `errno = ERANGE` (invalid key size).

#### 4.6.2.10 int cryptos\_md\_algo\_code (const char \* *md\_algo\_name*, int \* *code*)

Returns the GCRY code for the algorithm with *md\_algo\_name* name. If the specified name is NULL, returns the code for SHA1, which is the default message digest algorithm in this system.

##### Parameters:

- ← *md\_algo\_name* The name of the digest algorithm, in char\* representation.
- *code* Variable to store the corresponding code.

##### Returns:

The corresponding error code for integer returning functions, i.e., `I_CRYPTOS_OK` if no error was present and `I_CRYPTOS_ERR` if an error occurred with `errno` updated.

##### Return values:

- `I_CRYPTOS_OK` with `errno = 0` (No error).

#### 4.6.2.11 int parse\_packet (cryptos\_config\_t \* *cc*, byte \* *packet*, uint64\_t *p\_len*, byte \* *data*, uint64\_t \* *d\_len*, uint64\_t \* *r\_data*)

Parses the received data, returning the data in it. Parses the packet *packet*, of *p\_len* bytes, using the cryptos context defined by *cc*, verifying it's integrity and that matches with the expected emission and packet IDs. If everything goes OK, the recovered data will be stored in *data*, of *d\_len* allocated bytes, which will be reallocated (and *d\_len* updated consequently) if more space is needed. The total amount of pure data recovered, in bytes, will be stored in *r\_data*.

A packet will have the structure depicted below:

```

----- | SYNC | DATA_LENGTH | IV |
EMISSION_ID | PACKET_ID |.. DATA .. | DIGEST | -----
-----
```

- The field SYNC will be equal to CRYPTOS\_SYNC\_HEADER and will be of CRYPTOS\_SYNC\_HEADER\_LEN bytes long.
- The field DATA\_LENGTH will mark the length of the field DATA and will be of CRYPTOS\_LENGTH\_HEADER\_LEN bytes long.
- The field IV will be of CRYPTOS\_IV\_LEN bytes
- The field EMISSION\_ID will be CRYPTOS\_EMISSION\_LEN bytes long
- The field PACKET\_ID will be CRYPTOS\_PACKET\_LEN bytes long
- The field DATA will have the size marked in the DATA\_LENGTH field
- The DIGEST field will have a variable size, depending on the digest algorithm in use, and will vary from 24 bits (CRC24) to 64 bytes (SHA512) although high sizes are not likely to happen.

#### Parameters:

*cc* Cryptos context.  
*packet* Packet to parse  
*p\_len* Allocated size, in bytes, for *packet*  
*data* Array to store the recovered data in  
*d\_len* Allocated size, in bytes, for *data*  
*r\_data* Final successfully recovered amount of data, in bytes.

#### Returns:

The corresponding error code for integer returning functions, i.e., I\_CRYPTOS\_OK if no error was present and I\_CRYPTOS\_ERR if an error occurred with errno updated.

#### Return values:

**I\_CRYPTOS\_OK** with errno = 0 (No error).  
**I\_CRYPTOS\_ERR** with errno = EINVAL (invalid argument).  
**I\_CRYPTOS\_ERR** with errno != 0 && errno != EINVAL (see corresponding error code)

#### 4.6.2.12 int produce\_packet (cryptos\_config\_t \* cc, byte \* data, uint64\_t d\_len, byte \* packet, uint64\_t p\_len, uint64\_t \* w\_data)

Produces a new packet ready to be sent. Prepares a new packet ready to be sent. The data field will be ciphered and inserted in the packet. The header fields and integrity checks will be generated accordingly to the cryptos configuration received, and, upon successful execution, the packet id will be increased for the next packet.

A packet will have the structure depicted below:

```

----- | SYNC | DATA_LENGTH | IV |
EMISSION_ID | PACKET_ID |.. DATA .. | DIGEST | -----
-----

```

- The field SYNC will be equal to CRYPTOS\_SYNC\_HEADER and will be of CRYPTOS\_SYNC\_HEADER\_LEN bytes long.

- The field DATA\_LENGTH will mark the length of the field DATA and will be of CRYPTOS\_LENGTH\_HEADER\_LEN bytes long.
- The field IV will be of CRYPTOS\_IV\_LEN bytes
- The field EMISSION\_ID will be CRYPTOS\_EMISSION\_LEN bytes long
- The field PACKET\_ID will be CRYPTOS\_PACKET\_LEN bytes long
- The field DATA will have the size marked in the DATA\_LENGTH field and will be the only field ciphered.
- The DIGEST field will have a variable size, depending on the digest algorithm in use, and will vary from 24 bits (CRC24) to 64 bytes (SHA512) although high sizes are not likely to happen. The digest includes the whole packet except the SYNC field.

**Parameters:**

*cc* Cryptos config structure

*data* The (plain) data to write

*d\_len* The length of *data*, in bytes

*packet* The byte array in which store the result

*p\_len* The memory allocated for *packet*, in bytes.

*w\_data* The number of data bytes successfully included in the current packet. Note that the amount of received plain data can exceed the amount of data that can be included in the current packet.

**Returns:**

The corresponding error code for integer returning functions, i.e., I\_CRYPTOS\_OK if no error was present and I\_CRYPTOS\_ERR if an error occurred with *errno* updated.

**Return values:**

**I\_CRYPTOS\_OK** with *errno* = 0 (No error).

**I\_CRYPTOS\_ERR** with *errno* = EINVAL (invalid argument).

**I\_CRYPTOS\_ERR** with *errno* != 0 && *errno* != EINVAL (see corresponding error code)

## 4.7 lib/miscellaneous.c File Reference

## 4.8 lib/miscellaneous.h File Reference

```
#include "steganos_types.h"
#include "global_types.h"
```

### Defines

- `#define L_MISC_OK 0`  
*Error code for 'No error' to use in functions of integer return type.*
- `#define L_MISC_ERR 1`  
*Error code for 'Error occured' to use in functions of integer return type.*
- `#define L_MISC_INVALID_XML -1`  
*Error code for 'Invalid xml file' while validating an xml file.*
- `#define XML_KEY "key"`  
*String representing the field 'key' in xml config. files.*
- `#define XML_KEY_LENGTH "length"`  
*String representing the field 'length' within key elements in xml config. files.*
- `#define XML_KEY_BITSTREAM "bitstream"`  
*String representing the field 'bitstream' within key elements in xml config. files.*
- `#define XML_SYNCHROMODE "synchromode"`  
*String representing the field 'synchromode' in xml config. files.*
- `#define XML_HIDEMETHOD "hidemethod"`  
*String representing the field 'hidemethod' in xml config. files.*
- `#define XML_AGGRESS "aggressiveness"`  
*String representing the field 'agressiveness' in xml config. files.*
- `#define XML_LEAFS 5`  
*Number of "leaf nodes" (i.e. elements of type other than child) in the steganos DTD.*

### Functions

- `int to_byte (void *stream, size_t nmemb, size_t size, byte *byte_stream)`  
*Converts the given pointer of generic type to a byte pointer.*
- `int print_bytestream (int fd, byte *stream, int length)`  
*Prints a bytestream in fd.*
- `int bitstream_rot (byte *bitstream, int length, int rot)`  
*Rotates bitstream rot bits to the left.*

- int `bitstream_ror` (byte \*bitstream, int length, int rot)  
*Rotates bitstream rot bits to the right.*
- int `message_log` (const char \*caller, const char \*message)  
*Writes the specified message in the log file log\_<pid>.*
- int `zlib_def` (FILE \*source, FILE \*dest, int level)
- int `zlib_inf` (FILE \*source, FILE \*dest)
- void `zlib_zerr` (int ret)

## 4.8.1 Define Documentation

### 4.8.1.1 #define I\_MISC\_ERR 1

Error code for 'Error occurred' to use in functions of integer return type.

### 4.8.1.2 #define I\_MISC\_INVALID\_XML -1

Error code for 'Invalid xml file' while validating an xml file.

### 4.8.1.3 #define I\_MISC\_OK 0

Error code for 'No error' to use in functions of integer return type.

### 4.8.1.4 #define XML\_AGGRESS "aggressiveness"

String representing the field 'agressiveness' in xml config. files.

### 4.8.1.5 #define XML\_HIDEMETHOD "hidemethod"

String representing the field 'hidemethod' in xml config. files.

### 4.8.1.6 #define XML\_KEY "key"

String representing the field 'key' in xml config. files.

### 4.8.1.7 #define XML\_KEY\_BITSTREAM "bitstream"

String representing the field 'bitstream' within key elements in xml config. files.

### 4.8.1.8 #define XML\_KEY\_LENGTH "length"

String representing the field 'length' within key elements in xml config. files.

#### 4.8.1.9 #define XML\_LEAFS 5

Number of "leaf nodes" (i.e. elements of type other than child) in the steganos DTD.

#### 4.8.1.10 #define XML\_SYNCHROMODE "synchromode"

String representing the field 'synchromode' in xml config. files.

### 4.8.2 Function Documentation

#### 4.8.2.1 int bitstream\_rol (byte \* *bitstream*, int *length*, int *rot*)

Rotates bitstream rot bits to the left.

##### Parameters:

- ↔ *bitstream* The bit stream to rotate at the input, the rotated resulting bitstream at the output.
- ← *length* The length of bitstream, in bits.
- ← *rot* The number of bits to rotate to the left.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_MISC\_OK if no error was present and I\_MISC\_ERR if an error occurred with errno updated.

##### Return values:

- I\_MISC\_OK* with errno = 0 (No error).
- I\_MISC\_ERR* if unable to open log file, with errno updated to consequently (see fopen man page).

#### 4.8.2.2 int bitstream\_ror (byte \* *bitstream*, int *length*, int *rot*)

Rotates bitstream rot bits to the right.

##### Parameters:

- ↔ *bitstream* The bit stream to rotate at the input, the rotated resulting bitstream at the output.
- ← *length* The length of bitstream, in bits.
- ← *rot* The number of bits to rotate to the right.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_MISC\_OK if no error was present and I\_MISC\_ERR if an error occurred with errno updated.

##### Return values:

- I\_MISC\_OK* with errno = 0 (No error).
- I\_MISC\_ERR* if unable to open log file, with errno updated to consequently (see fopen man page).

#### 4.8.2.3 int message\_log (const char \* *caller*, const char \* *message*)

Writes the specified message in the log file log\_<pid>.

##### Parameters:

- ← *caller* Name of the calling function.
- ← *message* The message to write in the log.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_MISC\_OK if no error was present and I\_MISC\_ERR if an error occurred with errno updated.

##### Return values:

- I\_MISC\_OK* with errno = 0 (No error).
- I\_MISC\_ERR* if unable to open log file, with errno updated to consequently (see fopen man page).

#### 4.8.2.4 int print\_bytestream (int *fd*, byte \* *stream*, int *length*)

Prints a bytestream in fd.

##### Parameters:

- ← *fd* The file descriptor to print in. -1 to write in the system log
- ← *stream* The stream to print.
- ← *length* The length of *stream* in bytes.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_MISC\_OK if no error was present and I\_MISC\_ERR if an error occurred with errno updated.

##### Return values:

- I\_MISC\_OK* with errno = 0 (No error).
- I\_MISC\_ERR* with errno = EINVAL (Invalid argument).

#### 4.8.2.5 int to\_byte (void \* *stream*, size\_t *nmemb*, size\_t *size*, byte \* *byte\_stream*)

Converts the given pointer of generic type to a byte pointer. Converts the pointer *stream*, composed by *nmemb* elements of size *size* bytes each into a byte pointer, stored in *byte\_stream*. Uses big endian ordering.

##### Parameters:

- ← *stream* The void pointer to convert.
- ← *nmemb* The number of elements in *stream*.
- ← *size* The size, in bytes, of each *stream*'s element.
- ↔ *byte\_stream* The pointer which will store the byte conversion of *stream*. Enough memory must have been allocated previously to calling the function, which must be, at least nmemb\*size bytes.



**Returns:**

The corresponding error code for integer returning functions, i.e., `I_MISC_OK` if no error was present and `I_MISC_ERR` if an error occurred with `errno` updated.

**Return values:**

***I\_MISC\_OK*** with `errno = 0` (No error).

***I\_MISC\_ERR*** with `errno = EINVAL` (invalid argument).

**4.8.2.6** `int zlib_def (FILE * source, FILE * dest, int level)`

**4.8.2.7** `int zlib_inf (FILE * source, FILE * dest)`

**4.8.2.8** `void zlib_zerr (int ret)`

## 4.9 lib/numbers.c File Reference

## 4.10 lib/protocols.c File Reference

## 4.11 lib/protocols.h File Reference

```
#include "vorbis/codec.h"
#include "steganos_types.h"
#include "cryptos_types.h"
#include "global_types.h"
```

### Functions

- `int cryptos_forward (cryptos_config_t *cc, cryptos_protocol_buffer_t *cb, uint64_t data_size)`  
*Produces a new cryptographic packet when necessary.*
- `int cryptos_inverse (cryptos_config_t *cc, cryptos_protocol_buffer_t *clb)`  
*Tries to recover a new crypto packet from the cryptographic layer buffer.*
- `int steganos_vorbis_config_init (vorbis_config_t *vc, int rate, int pcmend, int mult, int *postlist, int *forward_index, int posts)`  
*Initializes the structure containing basic vorbis codec info needed in the steganographic layer.*
- `int steganos_vorbis_config_free (vorbis_config_t *vc)`  
*Frees the given `vorbis_config_t` structure.*
- `int steganos_forward (steganos_state_t *ss, vorbis_config_t *vc, int *floor, int *posts, float *residue, byte *buffer, size_t buffer_len, int *hided)`  
*Interface to the sender's steganographic functionality.*
- `int steganos_inverse (steganos_state_t *ss, vorbis_config_t *vc, int *posts, int *floor, float *residue, float sigma, byte *buffer, int buffer_sz, int *sd_read)`  
*Interface to the receiver's steganographic functionality.*

### 4.11.1 Function Documentation

#### 4.11.1.1 `int cryptos_forward (cryptos_config_t * cc, cryptos_protocol_buffer_t * cb, uint64_t data_size)`

Produces a new cryptographic packet when necessary. When this function is called, and the cryptographic layer buffer is at risk of underflow (i.e., if it has less than MAX\_SUBLIMINAL\_SIZE bits stores), produces a new crypto packet, storing it in the cryptographic layer buffer. The data to include is read from the file descriptor stored in the cryptographic layer buffer.

#### Parameters:

- ← **cc** Cryptographic layer configuration structure.
- ← **cb** Cryptographic layer buffer.
- ← **data\_size** The number of bits of data to include in the current crypto-packet. If a value of 0 is passed, or it exceeds the maximum data size per packet specified in *cc*, the default data size will be used instead.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

`I_STEGANOS_OK` with `errno = 0` (No error).

`I_STEGANOS_ERR` with `errno = EINVAL` (invalid argument).

**4.11.1.2 int cryptos\_inverse (cryptos\_config\_t \* cc, cryptos\_protocol\_buffer\_t \* cb)**

Tries to recover a new crypto packet from the cryptographic layer buffer. When called, this function explores the cryptographic layer buffer, using the context defined by the cryptographic configuration structure `cc`, and if a complete new packet is successfully parsed, writes the recovered data in the file descriptor stored in the cryptographic layer buffer.

**Parameters:**

← `cc` Cryptographic layer configuration structure.

← `cb` Cryptographic layer buffer.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

`I_STEGANOS_OK` with `errno = 0` (No error).

`I_STEGANOS_ERR` with `errno = EINVAL` (invalid argument).

**4.11.1.3 int steganos\_forward (steganos\_state\_t \* ss, vorbis\_config\_t \* vc, int \* floor, int \* posts, float \* residue, byte \* buffer, size\_t buffer\_len, int \* hided)**

Interface to the sender's steganographic functionality. Through this interface, a sender program will be able to hide in the floor and residue received vectors, from a Vorbis audio stream, the bitstream stored in the field indicated by the *input* field within the 'steganos\_config.xml' file. The steganographic layer will try to hide as much data as possible, and will update *sent* with the number of bits successfully hidden.

**Parameters:**

← `ss` The steganographic protocol internal structure

← `vc` Vorbis block and look info.

← `floor` The Vorbis audio floor vector.

↔ `posts` The Vorbis posts vector.

↔ `residue` The Vorbis audio residue vector.

← `buffer` Buffer containing the data to send.

← `buffer_len` Length of buffer, in bytes, containing initialized data.

↔ `hided` Pure data successfully hidden, in bits.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

***`I_STEGANOS_OK`*** with `errno = 0` (No error).

***`I_STEGANOS_ERR`*** with `errno = EINVAL` (invalid argument).

#### 4.11.1.4 `int steganos_inverse (steganos_state_t * ss, vorbis_config_t * vc, int * posts, int * floor, float * residue, float sigma, byte * buffer, int buffer_sz, int * sd_read)`

Interface to the receiver's steganographic functionality. Through this interface, a receiver program can unhide as much subliminal data as may have been hidden in the Vorbis audio *floor* and/or *residue* vectors, if any.

**Parameters:**

← ***`ss`*** The steganographic protocol internal structure

← ***`vc`*** Vorbis block and look info.

← ***`posts`*** The Vorbis audio floor posts vector.

← ***`floor`*** The Vorbis audio floor vector.

← ***`residue`*** The Vorbis audio residue vector.

← ***`sigma`*** Sigma parameter if ISS.

↔ ***`buffer`*** Buffer to store the subliminal data.

← ***`buffer_sz`*** Size of ***`buffer`***, in bytes.

→ ***`sd_read`*** Number of bytes recovered

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

***`I_STEGANOS_OK`*** with `errno = 0` (No error).

***`I_STEGANOS_ERR`*** with `errno = EINVAL` (invalid argument).

**See also:**

[steganos\\_state\\_t](#)

#### 4.11.1.5 `int steganos_vorbis_config_free (vorbis_config_t * vc)`

Frees the given [vorbis\\_config\\_t](#) structure.

**Parameters:**

← ***`vc`*** Structure to free

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

`I_STEGANOS_OK` with `errno = 0` (No error).

**4.11.1.6 int steganos\_vorbis\_config\_init (vorbis\_config\_t \* vc, int rate, int pcmend, int mult, int \* postlist, int \* forward\_index, int posts)**

Initializes the structure containing basic vorbis codec info needed in the steganographic layer. Initializes the structure containing basic vorbis codec info needed in the steganographic layer. It is very important to note that, unless the other \*\_init functions, this one DOES NOT allocate memory for the internal arrays, as they are just make point the Vorbis ones. Therefore, there exists no respective \*\_free function, as the inner variables are supposed to be freed by Vorbis code.

**Parameters:**

- ↔ *vc* Structure that will group all the essential Vorbis variables
- ← *rate* Sampling rate
- ← *pcmend* Length of the floor and residue vectors
- ← *mult* Multiplier used in the posts calculation
- ← *postlist* Array of posts
- ← *forward\_index* Array of posts X axis ordering
- ← *posts* Number of elements in the postlist array

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

`I_STEGANOS_OK` with `errno = 0` (No error).

`I_STEGANOS_ERR` with `errno = EINVAL` (invalid argument).

## 4.12 lib/steganos\_channel.c File Reference



## 4.13 lib/steganos\_channel.h File Reference

```
#include "global_types.h"
#include "steganos_types.h"
#include "numbers.h"
```

### Defines

- #define [DISTRESS\\_SYNC\\_BYTES](#) 3  
*Size, in bytes, of the "distress" header, used to resynchronize when using streaming mode.*
- #define [DISTRESS\\_SYNC\\_BITS](#) DISTRESS\_SYNC\_BYTES\*BITS\_PER\_BYTE  
*Size, in bits, of the "distress" header, used to resynchronize when using streaming mode.*
- #define [FILE\\_SIZE\\_BYTES](#) 4  
*Size, in bytes, of the field that will mark the amount of subliminal data that we will try to hide at a given audio file.*
- #define [FILE\\_SIZE\\_BITS](#) FILE\_SIZE\_BYTES\*BITS\_PER\_BYTE  
*Size, in bits, of the field that will mark the amount of subliminal data that we will try to hide at a given audio file.*
- #define [SYNCHRO\\_HEADER\\_BYTES\\_RES](#) 1  
*Length of the synchronization header field when using RES\_HEADER synchronization mode. Doesn't include the size field bytes.*
- #define [SIZE\\_FIELD\\_BITS](#) 8  
*Length of the synchronization field addressed to indicate the number of subliminal bits hided in a given frame and channel.*
- #define [RES\\_HEADER\\_BITS](#) SYNCHRO\_HEADER\_BYTES\_RES\*BITS\_PER\_BYTE+SIZE\_FIELD\_BITS  
*The size in bits of THE WHOLE header when using RES\_HEADER synchronization method.*
- #define [MAX\\_SUBLIMINAL\\_SIZE](#) ((1 << SIZE\_FIELD\_BITS) - 1)  
*Maximum number of subliminal bits that could be hidden in a residue.*
- #define [BITS\\_PARITY](#) 2  
*Number of bits to use to calculate the parity in the parity bit method.*
- #define [ITU\\_R\\_BS\\_468\\_SIZE](#) 20  
*Size of the global static array ITU\_R\_BS\_468.*

### Functions

- int [steganos\\_state\\_init](#) (steganos\_state\_t \*ss, int da, int [hide\\_method](#), int [sync\\_method](#), char \*key, int keylen)  
*Initializes the members of the received stegano\_state\_t structure.*

- int [steganos\\_state\\_reset\\_iter](#) ([steganos\\_state\\_t](#) \*ss)  
*Resets the members of the steganos state structure that change from one iteration to the next.*
- int [steganos\\_state\\_free](#) ([steganos\\_state\\_t](#) \*ss)  
*Frees the members of the received stegano\_state\_t structure. The structure itself must be freed separately.*
- int [steganos\\_prepare\\_packet\\_keys](#) ([vorbis\\_config\\_t](#) \*vc, [steganos\\_state\\_t](#) \*ss)  
*Manipulates the keys to be ready to be used in a new steganos packet.*
- int [steganos\\_free\\_packet\\_keys](#) ([steganos\\_state\\_t](#) \*ss)  
*Frees the keys used in the current steganos packet.*
- int [set\\_subliminal\\_capacity\\_limit](#) ([steganos\\_state\\_t](#) \*ss, float \*residue, const long rate, const int res\_len)  
*Determines and sets the subliminal capacity limit for the current frame and channel.*
- int [hide\\_data](#) ([steganos\\_state\\_t](#) \*ss, [byte](#) \*data, const int d\_len, int \*floor, float \*residue, const int res\_len, int \*size)  
*Determines the final usage of the residue subliminal channel for the current frame and channel by getting the desired size of subliminal data to hide in it.*
- int [unhide\\_data](#) ([steganos\\_state\\_t](#) \*ss, int \*floor, float \*residue, const int res\_len, [byte](#) \*\*data, int \*read)  
*Reads subliminal data from a subliminal residue.*
- int [parity\\_bits\\_method](#) ([byte](#) \*plain\_mess, int p\_m\_len, int \*floor, float \*res, int res\_len, [steganos\\_key\\_t](#) \*key, [byte](#) \*sub\_mess, int \*s\_m\_len, [prng\\_t](#) \*prng)  
*Implementation of Anderson's and Petitcolas's parity bit method.*
- int [void\\_method](#) ([byte](#) \*plain\_mess, int p\_m\_len, int \*floor, float \*res, int res\_len, [steganos\\_key\\_t](#) \*key, [byte](#) \*sub\_mess, int \*s\_m\_len, [prng\\_t](#) \*prng)  
*Applies no hiding method to the subliminal data.*
- int [synchro\\_iss](#) ([vorbis\\_config\\_t](#) \*vc, [steganos\\_key\\_t](#) \*key, int \*posts, float \*residue, void \*cfg, int decoding, int \*bit, [prng\\_t](#) \*prng)  
*Implementation of ISS watermarking for synchronization method.*
- int [iss\\_cfg\\_init](#) ([steganos\\_state\\_t](#) \*ss, int \*posts, int posts\_len, float sigma, [iss\\_cfg\\_t](#) \*iss\_cfg)  
*Initializes the sigma, lambda and alpha values needed for ISS accordingly to the formulas from Malvar and Florencio's.*
- int [iss\\_cfg\\_free](#) ([iss\\_cfg\\_t](#) \*iss\_cfg)  
*Frees the memory allocated for a [iss\\_cfg\\_t](#) structure.*
- int [iss\\_simulate\\_floor](#) ([vorbis\\_config\\_t](#) \*vc, int \*posts, int \*floor)  
*Simulates the floor that will be calculated by te receiver may the post vector be like posts.*
- int [desynchro\\_res\\_header](#) ([vorbis\\_config\\_t](#) \*vc, int \*res\_lineup, int \*posts, int \*floor, float \*residue, void \*cfg, [steganos\\_key\\_t](#) \*hiding\_key, [hide\\_method](#) hide, [prng\\_t](#) \*prng)  
*Type definition for desynchronization functions to use in the protocol.*

- int `desynchro_iss` (`vorbis_config_t` \*vc, int \*res\_lineup, int \*posts, int \*floor, float \*residue, void \*cfg, `steganos_key_t` \*hiding\_key, `hide_method` hide, `prng_t` \*prng)  
*Desynchronization function when using ISS synchronization method.*
- int `steganos_key_init` (`byte` \*byte\_key, const int key\_len, `steganos_key_t` \*key)  
*Allocates memory for the key fields and sets them to the received values.*
- int `calculate_residue_lineup` (`steganos_state_t` \*ss, const int res\_len)  
*Calculates the residue lineup using the prng in ss.*

### 4.13.1 Define Documentation

#### 4.13.1.1 #define BITS\_PARITY 2

Number of bits to use to calculate the parity in the parity bit method.

#### 4.13.1.2 #define DISTRESS\_SYNC\_BITS DISTRESS\_SYNC\_BYTES\*BITS\_PER\_BYTE

Size, in bits, of the "distress" header, used to resynchronize when using streaming mode.

#### 4.13.1.3 #define DISTRESS\_SYNC\_BYTES 3

Size, in bytes, of the "distress" header, used to resynchronize when using streaming mode.

#### 4.13.1.4 #define FILE\_SIZE\_BITS FILE\_SIZE\_BYTES\*BITS\_PER\_BYTE

Size, in bits, of the field that will mark the amount of subliminal data that we will try to hide at a given audio file.

#### 4.13.1.5 #define FILE\_SIZE\_BYTES 4

Size, in bytes, of the field that will mark the amount of subliminal data that we will try to hide at a given audio file.

#### 4.13.1.6 #define ITU\_R\_BS\_468\_SIZE 20

Size of the global static array ITU\_R\_BS\_468.

#### 4.13.1.7 #define MAX\_SUBLIMINAL\_SIZE ((1 << SIZE\_FIELD\_BITS) - 1)

Maximum number of subliminal bits that could be hidden in a residue.

#### 4.13.1.8 #define RES\_HEADER\_BITS SYNCHRO\_HEADER\_BYTES\_RES\*BITS\_PER\_BYTE+SIZE\_FIELD\_BITS

The size in bits of THE WHOLE header when using RES\_HEADER synchronization method.

#### 4.13.1.9 #define SIZE\_FIELD\_BITS 8

Length of the synchronization field addressed to indicate the number of subliminal bits hided in a given frame and channel.

#### 4.13.1.10 #define SYNCHRO\_HEADER\_BYTES\_RES 1

Length of the synchronization header field when using RES\_HEADER synchronization mode. Doesn't include the size field bytes.

### 4.13.2 Function Documentation

#### 4.13.2.1 calculate\_residue\_lineup (steganos\_state\_t \* ss, const int res\_len)

Calculates the residue lineup using the prng in ss.

##### Parameters:

- ← *ss* Steganos state structure.
- ← *res\_len* Length of the current residue vector.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated.

##### Return values:

*I\_STEGANOS\_OK*  
*I\_STEGANOS\_ERR*

#### 4.13.2.2 int desynchro\_iss (vorbis\_config\_t \* vc, int \* res\_lineup, int \* posts, int \* floor, float \* residue, void \* cfg, steganos\_key\_t \* hiding\_key, hide\_method hide, prng\_t \* prng)

Desynchronization function when using ISS synchronization method. This function should be called when it is not possible to hide information in a given frame-channel and the synchronization method in use is ISS. This function will set the first SIZE\_FIELD\_BITS of the given residue to 0, ensuring this way that, despite the receiver will receive a 1-marked floor, he'll read 0 bytes of data.

##### Parameters:

- ← *vc* Vorbis block and look info.
- ← *res\_lineup* Residue ordering
- ↔ *posts* Current floor posts vector.
- ← *floor* Floor vector
- ↔ *residue* Current residue vector. Will be changed if it comes "naturally" synchronized.
- ← *cfg* ISS configuration options (sigma, alpha, etc.)
- ← *hiding\_key* Key for hiding
- ← *hide* Hiding method (pointer to function) in use in the current frame and channel.

← *prng* PRNG in use

#### Returns:

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated.

#### Return values:

*I\_STEGANOS\_OK* with errno = 0 (No error).

*I\_STEGANOS\_ERR* with errno = EINVAL (invalid argument).

#### See also:

[synchro\\_method\\_et](#)  
iss\_synchronization

#### 4.13.2.3 int desynchro\_res\_header (vorbis\_config\_t \* vc, int \* res\_lineup, int \* posts, int \* floor, float \* residue, void \* cfg, steganos\_key\_t \* hiding\_key, hide\_method hide, prng\_t \* prng)

Type definition for desynchronization functions to use in the protocol. Functions of this type shall solve the false positive synchronization problem that may happen depending on the synchro\_method used. Depending on the specific method in use, the *floor* and *residue* vectors will or won't be used.

#### Parameters:

← *vc* Vorbis block and look info.  
 ← *res\_lineup* Residue ordering  
 ↔ *posts* Current floor posts vector.  
 ← *floor* Floor vector  
 ↔ *residue* Current residue vector. Will be changed if it comes "naturally" synchronized.  
 ← *cfg* This parameter will be used to pass any argument needed in a concrete synchronization method.  
 ← *hiding\_key* Key for hiding  
 ← *hide* Hiding method (pointer to function) in use in the current frame and channel.  
 ← *prng* PRNG in use

#### Returns:

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated.

#### Return values:

*I\_STEGANOS\_OK* with errno = 0 (No error).

*I\_STEGANOS\_ERR* with errno = EINVAL (invalid argument).

#### See also:

[synchro\\_method\\_et](#)  
iss\_synchronization

#### 4.13.2.4 `int hide_data (steganos_state_t * ss, byte * data, const int d_len, int * floor, float * residue, const int res_len, int * size)`

Determines the final usage of the residue subliminal channel for the current frame and channel by getting the desired size of subliminal data to hide in it. This function first retrieves the next 'ss->aggressiveness\*ss->max\_fc\_capacity' bits of subliminal data to hide in the current frame and channel residue, using the chosen hiding function. This let us make a more precise estimation of the final effects of the changes we will introduce in the original data. Once this is done, the nearest integer size of bits is hid in the residue vector, giving the final subliminal-residue vector. Note that this function shouldn't be called if ss->aggressiveness\*ss->max\_fc\_capacity is not big enough to store the meta-information required (plus some subliminal info).

##### Parameters:

- ↔ *ss* Internal state structure. The function updates the ss->protocol\_send\_state field, which must be SYNCHRONIZED at the input and will be CONTROLLED after a successful execution.
- ← *data* The remaining bitstream to hide. Note that can have greater, equal or less size than the subliminal channel usage.
- ← *d\_len* The length of the 'data' stream, in bits.
- ← *floor* The current frame and channel floor vecotr. Depending on the hiding method selected, it may or may not be used.
- ↔ *residue* The current frame and channel residue vector. After a successful execution of the function, it will store the subliminal residue.
- ← *res\_len* The length of the residue (and floor) vector.
- *size* The final number of bits of pure data (excluding headers) hid in the current frame and channel residue.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated

##### Return values:

- I\_STEGANOS\_OK* with errno = 0 (No error)
- I\_STEGANOS\_ERR* with errno = EINVAL (invalid argument)

##### See also:

[steganos\\_state\\_t](#)

#### 4.13.2.5 `int iss_cfg_free (iss_cfg_t * iss_cfg)`

Frees the memory allocated for a [iss\\_cfg\\_t](#) structure.

##### Parameters:

- ↔ *iss\_cfg* ISS configuration structure to free.

##### Returns:

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated.

**Return values:**

*I\_STEGANOS\_OK* with `errno = 0` (No error).

**See also:**

`iss_synchronization`

#### 4.13.2.6 `int iss_cfg_init (steganos_state_t * ss, int * posts, int posts_len, float sigma, iss_cfg_t * iss_cfg)`

Initializes the sigma, lambda and alpha values needed for ISS accordingly to the formulas from Malvar and Florencio's.

**Parameters:**

- ← *ss* Internal state structure.
- ← *posts* Post vector.
- ← *posts\_len* Number of elements in the posts vector.
- ← *sigma* Sigma parameter (watermark's strength)
- ↔ *iss\_cfg* ISS configuration structure.

**Returns:**

The corresponding error code for integer returning functions, i.e., *I\_STEGANOS\_OK* if no error was present and *I\_STEGANOS\_ERR* if an error occurred with `errno` updated.

**Return values:**

*I\_STEGANOS\_OK* with `errno = 0` (No error).  
*I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).

**See also:**

`iss_synchronization`

#### 4.13.2.7 `int iss_simulate_floor (vorbis_config_t * vc, int * posts, int * floor)`

Simulates the floor that will be calculated by the receiver may the post vector be like *posts*.

**Parameters:**

- ← *vc* Vorbis block and look configuration data.
- ↔ *floor* The floor vector, as will be calculated by the receiver. The memory must be allocated previously to calling the function.
- ← *posts* The posts vector.

**Returns:**

The corresponding error code for integer returning functions, i.e., *I\_STEGANOS\_OK* if no error was present and *I\_STEGANOS\_ERR* if an error occurred with `errno` updated.

**Return values:**

*I\_STEGANOS\_OK* with `errno = 0` (No error).  
*I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).

#### 4.13.2.8 `int parity_bits_method (byte * plain_mess, int p_m_len, int * floor, float * res, int res_len, steganos_key_t * key, byte * sub_mess, int * s_m_len, prng_t * prng)`

Implementation of Anderson's and Petitcolas's parity bit method. Implementation of Anderson's and Petitcolas's parity bit method. See "On The Limits of Steganosgraphy" by Anderson and Petitcolas. The function keeps the (\*hide\_method) typedef format.

##### Note:

This function uses the PRNG function `get_random_int`, therefore, the PRNG must have been seeded previously with the same key used for hiding in order to get the correct data from the residue/floor vectors.

##### Parameters:

- ← *plain\_mess* Plain message to hide.
- ← *p\_m\_len* Length of plain message in bits.
- ← *floor* Floor vector.
- ← *res* Residue vector.
- ← *res\_len* Number of elements in *floor* and *res* vectors.
- ← *key* Key to use.
- ↔ *sub\_mess* String in which the subliminal message will be stored. It has to be allocated previously to calling the function.
- ↔ *s\_m\_len* At the input, will store the allocated size (in bits) of *sub\_mess*. At the output, the length of the resulting sub\_message, in bits (it may be different than *p\_m\_len* as it may include de-synchronization bits), or discard them.
- ← *prng* PRNG in use.

##### Returns:

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

##### Return values:

- `I_STEGANOS_OK` with `errno = 0` (No error).
- `I_STEGANOS_ERR` with `errno = EINVAL` (invalid argument).

##### See also:

[hide\\_method](#)

#### 4.13.2.9 `int set_subliminal_capacity_limit (steganos_state_t * ss, float * residue, const long rate, const int res_len)`

Determines and sets the subliminal capacity limit for the current frame and channel. This function determines the subliminal capacity limit for the current frame and channel residue, updating the `steganos_state_t` internal state structure with the values obtained. This limits will be stored in the (float \*)[2] field as [-, +] maximum variations to introduce in the corresponding residual value of the frame and channel's residue vector. Also updates the maximum subliminal capacity for the whole frame and channel current residue. The protocol state must be IDLE at the input and MEASURED after a successful execution. Also initializes the `ss->res_mask` variable, which can be seen as some kind of "steganosgraphic mask", calculated initially from the maximum residual variations here obtained. The protocol state structure variables must have been allocated and reset previously.



**Parameters:**

- ↔ *ss* Internal state structure. It's field capacity\_limit will be updated in a successful execution. The protocol state must be IDLE at the input and MEASURED after a successful execution.
- ← *residue* The current frame and channel residue vector.
- ← *rate* The sampling rate used (in Hz), to obtain the frequency corresponding to each residue.
- ← *res\_len* Length of the residue and frequencies vectors.

**Returns:**

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated.

**Return values:**

- I\_STEGANOS\_OK* with errno = 0 (No error).
- I\_STEGANOS\_ERR* with errno = EINVAL (invalid argument).

**See also:**

[steganos\\_state\\_t](#)

**4.13.2.10 static int steganos\_free\_packet\_keys (steganos\_state\_t \* ss)**

Frees the keys used in the current steganos packet.

**Parameters:**

- ← *ss* Steganos state structure

**Returns:**

I\_STEGANOS\_OK

**4.13.2.11 int steganos\_key\_init (byte \* byte\_key, const int key\_len, steganos\_key\_t \* key)**

Allocates memory for the key fields and sets them to the received values. Allocates memory for the key fields and sets them to the received values.

**Parameters:**

- ← *byte\_key* The key value.
- ← *key\_len* The length of the *byte\_key* argument, in bits.
- ↔ *key* A pointer to a previously allocated [steganos\\_key\\_t](#) variable in which we'll store the key's members.

**Returns:**

The corresponding error code for integer returning functions, i.e., I\_STEGANOS\_OK if no error was present and I\_STEGANOS\_ERR if an error occurred with errno updated.

**Return values:**

- I\_STEGANOS\_OK* with errno = 0 (No error).
- I\_STEGANOS\_ERR* with errno = EINVAL (invalid argument).
- I\_STEGANOS\_ERR* with errno = ERANGE (invalid key size).

#### 4.13.2.12 `int steganos_prepare_packet_keys (vorbis_config_t * vc, steganos_state_t * ss)`

Manipulates the keys to be ready to be used in a new steganos packet. The key used to hide/sync is obtained following this flow: 1) Cipher md5(vc->forward\_index) with ARCFOUR, using the master key 2) Use the resulting ciphertext as frame specific keys

##### Parameters:

- ← *vc* Vorbis config structure
- ↔ *ss* Steganos state structure. The internal variables *ss*->hiding\_key and *ss*->synchro\_key will be updated upon successful execution.

##### Returns:

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

##### Return values:

- I\_STEGANOS\_OK* with `errno` = 0 (No error).
- I\_STEGANOS\_ERR* with `errno` = `EINVAL` (invalid argument).

#### 4.13.2.13 `int steganos_state_free (steganos_state_t * ss)`

Frees the members of the received `stegano_state_t` structure. The structure itself must be freed separately.

##### Parameters:

- ← *ss* Pointer to the steganographic protocol state variable.

##### Returns:

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

##### Return values:

- I\_STEGANOS\_OK* with `errno` = 0 (No error).
- I\_STEGANOS\_ERR* with `errno` = `EINVAL` (invalid argument).

##### See also:

[steganos\\_state\\_t](#)

#### 4.13.2.14 `int steganos_state_init (steganos_state_t * ss, int da, int hide_method, int sync_method, char * key, int keylen)`

Initializes the members of the received `stegano_state_t` structure. This function prepares the steganographic protocol for a new whole step, allocating and initializing the steganos state structure to the default values or those specified as arguments.

##### Parameters:

- ← *ss* Pointer to the steganographic protocol state variable.

- ← **da** Desired aggressiveness, i.e., the share of subliminal capacity to use, in 1/10 (e.g. 2 means use 20%)
- ← **hide\_method** Hide method to use, must be one of the defined in `hide_method_et`
- ← **sync\_method** Sync method to use, must be one of the defined in `sync_method_et`
- ← **key** Master key to use for deriving subkeys
- ← **keylen** Length of *key* in bits. Must be at least 128.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

- `I_STEGANOS_OK` with `errno = 0` (No error).
- `I_STEGANOS_ERR` with `errno = EINVAL` (invalid argument).

**See also:**

[steganos\\_state\\_t](#)

**4.13.2.15 int steganos\_state\_reset\_iter (steganos\_state\_t \* ss)**

Resets the members of the steganos state structure that change from one iteration to the next.

**Parameters:**

- ← **ss** Pointer to the steganographic protocol state variable.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

- `I_STEGANOS_OK` with `errno = 0` (No error).
- `I_STEGANOS_ERR` with `errno = EINVAL` (invalid argument).

**See also:**

[steganos\\_state\\_t](#)

**4.13.2.16 int synchro\_iss (vorbis\_config\_t \* vc, steganos\_key\_t \* key, int \* posts, float \* residue, void \* cfg, int decoding, int \* bit, prng\_t \* prng)**

Implementation of ISS watermarking for synchronization method. Implements the Improved Spread Spectrum watermarking technique introduced by Malvar and Florencio (in "Improved Spread Spectrum: a new modulation technique for robust watermarking") to synchronize sender and receiver by marking the floor vector.

**Parameters:**

- ← *vc* Vorbis block and look information.
- ← *key* Key to use
- ↔ *posts* Current floor posts vector. After a successful execution, it will be modified as needed.
- ↔ *residue* Current residue vector. Not used here.
- ← *cfg* Pointer to a structure of [iss\\_cfg\\_t](#) with the alpha, lambda and sigma parameters needed for ISS.
- ← *decoding* Active if the function is called from the decoder's side,
- ↔ *bit* At the input, the desired bit to include if we are at the encoder's side; at the output, the received bit if we are at the decoder's side (i.e. 1 if we're telling the receiver there is hidden data and 0 if not).
- ← *prng* PRNG in use.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

- I\_STEGANOS\_OK* with `errno = 0` (No error).
- I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).
- I\_STEGANOS\_SYNC\_FAIL* with `errno = 0` (Unable to synchronize).

**See also:**

[synchro\\_method\\_et](#)  
[iss\\_synchronization](#)

#### 4.13.2.17 `int unhide_data (steganos_state_t * ss, int * floor, float * residue, const int res_len, byte ** data, int * read)`

Reads subliminal data from a subliminal residue. This function recovers the subliminal data hidden in the current residue. If the ISS synchronization mode is in use, and this function is called, means the residue surely shelters subliminal data. If the `RES_HEADER` synchro mode is in use, we have to look for the `SYNCHRO_HEADER` first, which will tell us if we have to recover any data. At the end of a successful execution, *data* will point to the recovered data and *read* store the number of bits read. If no data has been recovered, *read* will be 0.

**Note:**

This function uses the PRNG function *get\_random\_int*, therefore, the PRNG must have been seeded previously with the same key used for hiding in order to get the correct data from the residue/floor vectors.

**Parameters:**

- ↔ *ss* Internal state structure. The protocol state must be `ANALYZED` at the input and `READ` after a successful execution.
- ← *floor* The floor vector of the current frame and channel.
- ← *residue* The residue vector of the current frame and channel, containing the subliminal data, if any.

- ← *res\_len* The length of the residue and floor vectors.
- ↔ *data* Pointer to the bitstream which will store the subliminal data read after a successful execution.
- ↔ *read* Will store the final amount of data that has been possible to read, in bits.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

- I\_STEGANOS\_OK* with `errno = 0` (No error).
- I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).

**See also:**

[steganos\\_state\\_t](#)

#### 4.13.2.18 `int void_method (byte * plain_mess, int p_m_len, int * floor, float * res, int res_len, steganos_key_t * key, byte * sub_mess, int * s_m_len, prng_t * prng)`

Applies no hiding method to the subliminal data. This hiding method simply returns the first `p_m_len` bits of `plain_mess`.

**Parameters:**

- ← *plain\_mess* Plain message to hide.
- ← *p\_m\_len* Length of plain message in bits.
- ← *floor* Floor vector.
- ← *res* Residue vector.
- ← *res\_len* Number of elements in *floor* and *res* vectors.
- ← *key* Key to use.
- ↔ *sub\_mess* String in which the subliminal message will be stored. It has to be allocated previously to calling the function.
- ↔ *s\_m\_len* At the input, will store the allocated size (in bits) of *sub\_mess*. At the output, the length of the resulting sub\_message, in bits (it may be different than `p_m_len` as it may include de-synchronization bits).
- ← *prng* PRNG in use. Here, `NULL`.

**Returns:**

The corresponding error code for integer returning functions, i.e., `I_STEGANOS_OK` if no error was present and `I_STEGANOS_ERR` if an error occurred with `errno` updated.

**Return values:**

- I\_STEGANOS\_OK* with `errno = 0` (No error).
- I\_STEGANOS\_ERR* with `errno = EINVAL` (invalid argument).

**See also:**

[hide\\_method](#)

# Index

- aligned
  - steganos\_state\_t, 14
- alpha
  - iss\_cfg\_t, 10
- BITS\_PARITY
  - steganos\_channel.h, 57
- BITS\_PER\_BYTE
  - global\_types.h, 21
- bitstream\_rol
  - miscellaneous.h, 45
- bitstream\_ror
  - miscellaneous.h, 45
- buffer
  - cryptos\_protocol\_buffer\_t, 9
- buffer\_size
  - cryptos\_protocol\_buffer\_t, 9
- buffer\_used
  - cryptos\_protocol\_buffer\_t, 9
- byte
  - global\_types.h, 21
- calculate\_residue\_lineup
  - steganos\_channel.h, 58
- chd
  - cryptos\_config\_t, 5
- cipher\_algo
  - cryptos\_config\_t, 5
- cryptos\_buffer\_free
  - cryptos\_channel.h, 35
- cryptos\_buffer\_init
  - cryptos\_channel.h, 36
- CRYPTOS\_BUFFER\_MAX\_SIZE
  - cryptos\_channel.h, 34
- cryptos\_channel.h
  - cryptos\_buffer\_free, 35
  - cryptos\_buffer\_init, 36
  - CRYPTOS\_BUFFER\_MAX\_SIZE, 34
  - cryptos\_cipher\_algo\_code, 36
  - cryptos\_config\_free, 36
  - cryptos\_config\_init, 37
  - cryptos\_config\_set\_iv, 37
  - cryptos\_config\_set\_key, 38
  - CRYPTOS\_DEFAULT\_DATA\_SIZE, 34
  - CRYPTOS\_EMISSION\_HEADER\_LEN, 34
  - CRYPTOS\_IV\_HEADER\_LEN, 35
  - cryptos\_key\_free, 38
  - cryptos\_key\_init, 38
  - CRYPTOS\_LENGTH\_HEADER\_LEN, 35
  - CRYPTOS\_MAX\_DATA, 35
  - CRYPTOS\_MAX\_DIGEST\_LEN, 35
  - cryptos\_md\_algo\_code, 39
  - CRYPTOS\_MIN\_DIGEST\_LEN, 35
  - CRYPTOS\_PACKET\_HEADER\_LEN, 35
  - CRYPTOS\_SYNC\_HEADER\_LEN, 35
  - parse\_packet, 39
  - produce\_packet, 40
  - RATIO\_DD, 35
- cryptos\_cipher\_algo\_code
  - cryptos\_channel.h, 36
- cryptos\_config\_free
  - cryptos\_channel.h, 36
- cryptos\_config\_init
  - cryptos\_channel.h, 37
- cryptos\_config\_set\_iv
  - cryptos\_channel.h, 37
- cryptos\_config\_set\_key
  - cryptos\_channel.h, 38
- cryptos\_config\_t, 5
  - chd, 5
  - cipher\_algo, 5
  - default\_data\_size, 6
  - emission, 6
  - hmac, 6
  - iv, 6
  - key, 6
  - max\_data, 6
  - md\_algo, 6
  - md\_len, 6
  - mdhd, 6
  - packet, 6
- CRYPTOS\_DEFAULT\_DATA\_SIZE
  - cryptos\_channel.h, 34
- CRYPTOS\_EMISSION\_HEADER\_LEN
  - cryptos\_channel.h, 34
- cryptos\_forward
  - protocols.h, 50
- cryptos\_inverse
  - protocols.h, 51
- CRYPTOS\_IV\_HEADER\_LEN

- cryptos\_channel.h, 35
- cryptos\_key\_free
  - cryptos\_channel.h, 38
- cryptos\_key\_init
  - cryptos\_channel.h, 38
- cryptos\_key\_t, 7
  - key, 7
  - length, 7
- CRYPTOS\_LENGTH\_HEADER\_LEN
  - cryptos\_channel.h, 35
- CRYPTOS\_MAX\_DATA
  - cryptos\_channel.h, 35
- CRYPTOS\_MAX\_DIGEST\_LEN
  - cryptos\_channel.h, 35
- cryptos\_md\_algo\_code
  - cryptos\_channel.h, 39
- CRYPTOS\_MIN\_DIGEST\_LEN
  - cryptos\_channel.h, 35
- CRYPTOS\_PACKET\_HEADER\_LEN
  - cryptos\_channel.h, 35
- cryptos\_protocol\_buffer, 8
- cryptos\_protocol\_buffer\_t, 9
  - buffer, 9
  - buffer\_size, 9
  - buffer\_used, 9
  - fd, 9
  - offset, 9
- CRYPTOS\_SYNC\_HEADER\_LEN
  - cryptos\_channel.h, 35
- cryptos\_types.h
  - I\_CRYPTOS\_CHECK\_FAIL, 19
  - I\_CRYPTOS\_ERR, 19
  - I\_CRYPTOS\_OK, 20
- da
  - steganos\_state\_t, 14
- default\_data\_size
  - cryptos\_config\_t, 6
- desync
  - steganos\_state\_t, 14
- desync\_method
  - steganos\_types.h, 29
- desynchro\_iss
  - steganos\_channel.h, 58
- desynchro\_res\_header
  - steganos\_channel.h, 59
- desynchronize
  - steganos\_state\_t, 14
- DIRECT\_HIDING
  - steganos\_types.h, 31
- DISTRESS\_SYNC\_BITS
  - steganos\_channel.h, 57
- DISTRESS\_SYNC\_BYTES
  - steganos\_channel.h, 57
- emission
  - cryptos\_config\_t, 6
- fd
  - cryptos\_protocol\_buffer\_t, 9
- FILE\_SIZE\_BITS
  - steganos\_channel.h, 57
- FILE\_SIZE\_BYTES
  - steganos\_channel.h, 57
- FORCED\_RES\_HEADER
  - steganos\_types.h, 31
- forward\_index
  - vorbis\_config\_t, 17
- global\_types.h
  - BITS\_PER\_BYTE, 21
  - byte, 21
  - I\_ERR, 21
  - I\_OK, 21
- hide
  - steganos\_state\_t, 14
- hide\_data
  - steganos\_channel.h, 59
- hide\_method
  - steganos\_state\_t, 14
  - steganos\_types.h, 29
- hide\_method\_et
  - steganos\_types.h, 31
- hiding\_key
  - steganos\_state\_t, 14
- hmac
  - cryptos\_config\_t, 6
- I\_CRYPTOS\_CHECK\_FAIL
  - cryptos\_types.h, 19
- I\_CRYPTOS\_ERR
  - cryptos\_types.h, 19
- I\_CRYPTOS\_OK
  - cryptos\_types.h, 20
- I\_ERR
  - global\_types.h, 21
- I\_MISC\_ERR
  - miscellaneous.h, 44
- I\_MISC\_INVALID\_XML
  - miscellaneous.h, 44
- I\_MISC\_OK
  - miscellaneous.h, 44
- I\_OK
  - global\_types.h, 21
- I\_STEGANOS\_ERR
  - steganos\_types.h, 28
- I\_STEGANOS\_FRAME\_SKIP
  - steganos\_types.h, 28

- I\_STEGANOS\_OK
  - steganos\_types.h, 28
- I\_STEGANOS\_SYNC\_FAIL
  - steganos\_types.h, 28
- include/cryptos\_types.h, 19
- include/global\_types.h, 21
- include/numbers.h, 22
- include/steganos\_types.h, 27
- INDETERMINATE\_MARK
  - steganos\_types.h, 28
- ISS
  - steganos\_types.h, 31
- iss\_cfg\_free
  - steganos\_channel.h, 60
- iss\_cfg\_init
  - steganos\_channel.h, 61
- iss\_cfg\_t, 10
  - alpha, 10
  - lambda, 10
  - sigma, 10
  - u, 10
  - u\_norm, 10
- iss\_simulate\_floor
  - steganos\_channel.h, 61
- iters
  - prng\_t, 11
  - steganos\_state\_t, 14
- ITU\_R\_BS\_468\_SIZE
  - steganos\_channel.h, 57
- iv
  - cryptos\_config\_t, 6
- key
  - cryptos\_config\_t, 6
  - cryptos\_key\_t, 7
  - steganos\_key\_t, 12
- lambda
  - iss\_cfg\_t, 10
- length
  - cryptos\_key\_t, 7
  - steganos\_key\_t, 12
- lib/cryptos\_channel.c, 32
- lib/cryptos\_channel.h, 33
- lib/miscellaneous.c, 42
- lib/miscellaneous.h, 43
- lib/numbers.c, 48
- lib/protocols.c, 49
- lib/protocols.h, 50
- lib/steganos\_channel.c, 54
- lib/steganos\_channel.h, 55
- linear\_interpolation\_y
  - numbers.h, 22
- master\_key
  - steganos\_state\_t, 14
- max\_data
  - cryptos\_config\_t, 6
- max\_fc\_capacity
  - steganos\_state\_t, 14
- MAX\_SUBLIMINAL\_SIZE
  - steganos\_channel.h, 57
- md\_algo
  - cryptos\_config\_t, 6
- md\_len
  - cryptos\_config\_t, 6
- mdhd
  - cryptos\_config\_t, 6
- message\_log
  - miscellaneous.h, 45
- metadata\_sent
  - steganos\_state\_t, 14
- min\_fc\_capacity
  - steganos\_state\_t, 15
- miscellaneous.h
  - bitstream\_rol, 45
  - bitstream\_ror, 45
  - I\_MISC\_ERR, 44
  - I\_MISC\_INVALID\_XML, 44
  - I\_MISC\_OK, 44
  - message\_log, 45
  - print\_bytestream, 46
  - to\_byte, 46
  - XML\_AGGRESS, 44
  - XML\_HIDEMETHOD, 44
  - XML\_KEY, 44
  - XML\_KEY\_BITSTREAM, 44
  - XML\_KEY\_LENGTH, 44
  - XML\_LEAFS, 44
  - XML\_SYNCHROMODE, 45
  - zlib\_def, 47
  - zlib\_inf, 47
  - zlib\_zerr, 47
- mult
  - vorbis\_config\_t, 17
- numbers.h
  - linear\_interpolation\_y, 22
  - prng\_free, 23
  - prng\_get\_random\_int, 23
  - prng\_init, 23
  - prng\_reset\_byte, 24
  - prng\_set\_seed\_byte, 24
  - prng\_set\_seed\_int, 25
  - seek, 25
  - seek\_interval, 25
- offset



- cryptos\_protocol\_buffer\_t, 9
- out
  - steganos\_state\_t, 15
- packet
  - cryptos\_config\_t, 6
- PARITY\_BITS
  - steganos\_types.h, 31
- parity\_bits\_method
  - steganos\_channel.h, 61
- parse\_packet
  - cryptos\_channel.h, 39
- pcmend
  - vorbis\_config\_t, 17
- postlist
  - vorbis\_config\_t, 17
- posts\_len
  - vorbis\_config\_t, 17
- posts\_mode
  - steganos\_state\_t, 15
- print\_bytestream
  - miscellaneous.h, 46
- prng
  - steganos\_state\_t, 15
- prng\_free
  - numbers.h, 23
- prng\_get\_random\_int
  - numbers.h, 23
- prng\_init
  - numbers.h, 23
- prng\_reset\_byte
  - numbers.h, 24
- prng\_set\_seed\_byte
  - numbers.h, 24
- prng\_set\_seed\_int
  - numbers.h, 25
- prng\_t, 11
  - iters, 11
  - seed, 11
- produce\_packet
  - cryptos\_channel.h, 40
- protocols.h
  - cryptos\_forward, 50
  - cryptos\_inverse, 51
  - steganos\_forward, 51
  - steganos\_inverse, 52
  - steganos\_vorbis\_config\_free, 52
  - steganos\_vorbis\_config\_init, 53
- ra
  - steganos\_state\_t, 15
- rate
  - vorbis\_config\_t, 17
- RATIO\_DD
  - cryptos\_channel.h, 35
- read
  - steganos\_state\_t, 15
- remaining
  - steganos\_state\_t, 15
- RES\_HEADER
  - steganos\_types.h, 31
- RES\_HEADER\_BITS
  - steganos\_channel.h, 57
- res\_lineup
  - steganos\_state\_t, 15
- res\_max\_capacity
  - steganos\_state\_t, 15
- res\_min\_capacity
  - steganos\_state\_t, 15
- res\_occupied
  - steganos\_state\_t, 16
- seed
  - prng\_t, 11
- seek
  - numbers.h, 25
- seek\_interval
  - numbers.h, 25
- sent
  - steganos\_state\_t, 16
- set\_subliminal\_capacity\_limit
  - steganos\_channel.h, 62
- sigma
  - iss\_cfg\_t, 10
- SIZE\_FIELD\_BITS
  - steganos\_channel.h, 57
- status
  - steganos\_state\_t, 16
- steganos\_types.h
  - DIRECT\_HIDING, 31
  - FORCED\_RES\_HEADER, 31
  - ISS, 31
  - PARITY\_BITS, 31
  - RES\_HEADER, 31
- steganos\_channel.h
  - BITS\_PARITY, 57
  - calculate\_residue\_lineup, 58
  - desynchro\_iss, 58
  - desynchro\_res\_header, 59
  - DISTRESS\_SYNC\_BITS, 57
  - DISTRESS\_SYNC\_BYTES, 57
  - FILE\_SIZE\_BITS, 57
  - FILE\_SIZE\_BYTES, 57
  - hide\_data, 59
  - iss\_cfg\_free, 60
  - iss\_cfg\_init, 61
  - iss\_simulate\_floor, 61
  - ITU\_R\_BS\_468\_SIZE, 57

- MAX\_SUBLIMINAL\_SIZE, 57
- parity\_bits\_method, 61
- RES\_HEADER\_BITS, 57
- set\_subliminal\_capacity\_limit, 62
- SIZE\_FIELD\_BITS, 57
- steganos\_free\_packet\_keys, 63
- steganos\_key\_init, 63
- steganos\_prepare\_packet\_keys, 63
- steganos\_state\_free, 64
- steganos\_state\_init, 64
- steganos\_state\_reset\_iter, 65
- SYNCHRO\_HEADER\_BYTES\_RES, 58
- synchro\_iss, 65
- unhide\_data, 66
- void\_method, 67
- steganos\_forward
  - protocols.h, 51
- steganos\_free\_packet\_keys
  - steganos\_channel.h, 63
- steganos\_inverse
  - protocols.h, 52
- steganos\_key\_init
  - steganos\_channel.h, 63
- steganos\_key\_t, 12
  - key, 12
  - length, 12
- steganos\_prepare\_packet\_keys
  - steganos\_channel.h, 63
- steganos\_state\_free
  - steganos\_channel.h, 64
- steganos\_state\_init
  - steganos\_channel.h, 64
- steganos\_state\_reset\_iter
  - steganos\_channel.h, 65
- steganos\_state\_t, 13
  - aligned, 14
  - da, 14
  - desync, 14
  - desynchronize, 14
  - hide, 14
  - hide\_method, 14
  - hiding\_key, 14
  - iters, 14
  - master\_key, 14
  - max\_fc\_capacity, 14
  - metadata\_sent, 14
  - min\_fc\_capacity, 15
  - out, 15
  - posts\_mode, 15
  - prng, 15
  - ra, 15
  - read, 15
  - remaining, 15
  - res\_lineup, 15
  - res\_max\_capacity, 15
  - res\_min\_capacity, 15
  - res\_occupied, 16
  - sent, 16
  - status, 16
  - synchro\_key, 16
  - synchro\_method, 16
  - synchronize, 16
  - total\_sub\_capacity, 16
  - variation\_limit, 16
  - vc, 16
- steganos\_types.h
  - desync\_method, 29
  - hide\_method, 29
  - hide\_method\_et, 31
  - I\_STEGANOS\_ERR, 28
  - I\_STEGANOS\_FRAME\_SKIP, 28
  - I\_STEGANOS\_OK, 28
  - I\_STEGANOS\_SYNC\_FAIL, 28
  - INDETERMINATE\_MARK, 28
  - sync\_method, 30
  - synchro\_method\_et, 31
  - VIF\_POSIT, 28
  - VORBIS\_MAX\_BLOCK, 28
- steganos\_vorbis\_config\_free
  - protocols.h, 52
- steganos\_vorbis\_config\_init
  - protocols.h, 53
- sync\_method
  - steganos\_types.h, 30
- SYNCHRO\_HEADER\_BYTES\_RES
  - steganos\_channel.h, 58
- synchro\_iss
  - steganos\_channel.h, 65
- synchro\_key
  - steganos\_state\_t, 16
- synchro\_method
  - steganos\_state\_t, 16
- synchro\_method\_et
  - steganos\_types.h, 31
- synchronize
  - steganos\_state\_t, 16
- to\_byte
  - miscellaneous.h, 46
- total\_sub\_capacity
  - steganos\_state\_t, 16
- u
  - iss\_cfg\_t, 10
- u\_norm
  - iss\_cfg\_t, 10
- unhide\_data
  - steganos\_channel.h, 66

- variation\_limit
  - steganos\_state\_t, [16](#)
- vc
  - steganos\_state\_t, [16](#)
- VIF\_POSIT
  - steganos\_types.h, [28](#)
- void\_method
  - steganos\_channel.h, [67](#)
- vorbis\_config\_t, [17](#)
  - forward\_index, [17](#)
  - mult, [17](#)
  - pcmend, [17](#)
  - postlist, [17](#)
  - posts\_len, [17](#)
  - rate, [17](#)
- VORBIS\_MAX\_BLOCK
  - steganos\_types.h, [28](#)
- XML\_AGGRESS
  - miscellaneous.h, [44](#)
- XML\_HIDEMETHOD
  - miscellaneous.h, [44](#)
- XML\_KEY
  - miscellaneous.h, [44](#)
- XML\_KEY\_BITSTREAM
  - miscellaneous.h, [44](#)
- XML\_KEY\_LENGTH
  - miscellaneous.h, [44](#)
- XML\_LEAFS
  - miscellaneous.h, [44](#)
- XML\_SYNCHROMODE
  - miscellaneous.h, [45](#)
- zlib\_def
  - miscellaneous.h, [47](#)
- zlib\_inf
  - miscellaneous.h, [47](#)
- zlib\_zerr
  - miscellaneous.h, [47](#)