GenericCryptanalyzer

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

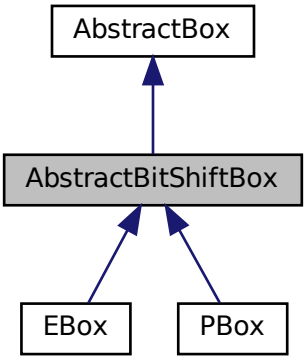
Class Documentation

4.1 AbstractBitShiftBox Class Reference

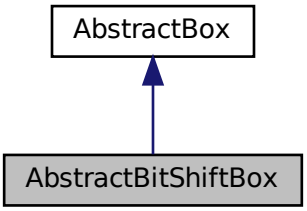
An AbstractBitShiftBox represents a generic box that transforms input in output by shifting bits around.

`#include <abstractbitshiftbox.h>`

Inheritance diagram for AbstractBitShiftBox:



Collaboration diagram for AbstractBitShiftBox:



Public Member Functions

- AbstractBitShiftBox (size_t in_size, size_t out_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes, const vector< size_t > &bit_src)
- AbstractBitShiftBox (size_t in_size, size_t out_size, const vector< size_t > &bit_src)

  similar to the previous constructor, but leaves `dst_boxes` *empty*
- void determine_next () override

  since apply_transformation is a linear deterministic transformations, this will set `is_det` *to* `true` *and* `prob` *to* `1`

Protected Member Functions

- void apply_transformation ()

  computes value of `out_bits` *from* `in_bits` *and* `bit_src`

Protected Attributes

- vector< size_t > bit_src

  an array which describes what bit from the input corresponds to each bit from the output

4.1.1 Detailed Description

An AbstractBitShiftBox represents a generic box that transforms input in output by shifting bits around.

More precisely, it takes a `vector<size_t> bit_src` as input and computes `out_bits[i] = in_bits[bit_src[i]]`

This class is an abstraction that encapsulates the functionality of both PBox and EBox.

See also

PBox

EBox

@inherits AbstractBox

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AbstractBitShiftBox() [1/2]

```
AbstractBitShiftBox::AbstractBitShiftBox (
            size_t in_size,
            size_t out_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes,
            const vector< size_t > & bit_src )
```

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
dst_boxes	output flow connections to following boxes
bit_src	an array used to compute `out_bits` from `in_bits`

Precondition

bit_source.size() == output_size

4.1.2.2 AbstractBitShiftBox() [2/2]

```
AbstractBitShiftBox::AbstractBitShiftBox (
            size_t in_size,
            size_t out_size,
            const vector< size_t > & bit_src )
```

similar to the previous constructor, but leaves `dst_boxes` empty

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
bit_src	an array used to compute `out_bits` from `in_bits`

Precondition

bit_source.size() == output_size

The documentation for this class was generated from the following files:

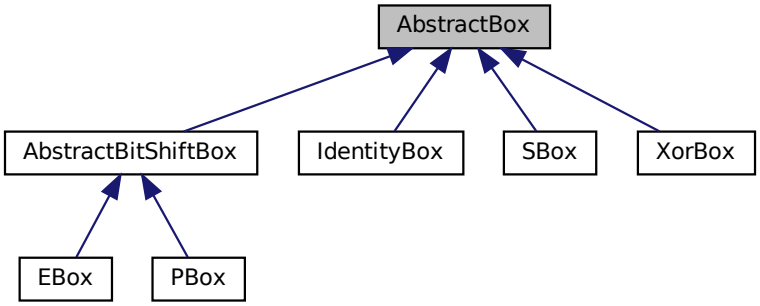
- src/box/abstractbitshiftbox.h
- src/box/abstractbitshiftbox.cpp

4.2 AbstractBox Class Reference

An AbstractBox represents an abstract idea of a block cipher component such as a Pbox, Sbox, Xor, Addition, etc. A cipher is composed of multiple such `boxes` that communicate with each other through connections.

`#include <abstractbox.h>`

Inheritance diagram for AbstractBox:



Public Member Functions

- AbstractBox (size_t in_size, size_t out_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes)
- AbstractBox (size_t in_size, size_t out_size)

  similar to the previous constructor, but leaves `dst_boxes` empty
- void add_dest (AbstractBoxPtr dst_box, BitsRange out_range, BitsRange in_range)

  add_dest adds a new destination box for the output of the box to flow to
- const dynamic_bitset & get_input ()

  getter for `in_bits`
- const dynamic_bitset & get_output ()

  getter for `out_bits`
- size_t input_size ()

  getter for the size of `in_bits`
- size_t output_size ()

  getter for the size of `out_bits`
- bool is_determined ()

  getter for `is_det`
- virtual void set_input (dynamic_bitset<> bits, const BitsRange &rng)

  sets a subrange `rng` of the input to the value of `bits`
- void notify_all ()

  notifies all the destination boxes after the output of the box is determined
- virtual void determine_next ()=0

  method to determine the next best output sorted by probabilities, if all possible outputs have been determined, `is_det` will be set to `true`
- virtual void reset_determination ()

  set the process to be undetermined by setting `is_det` to `false`
- double get_probability ()

  get the probability of the current characteristic

Protected Attributes

- dynamic_bitset in_bits

  the bits that flow into the box
- dynamic_bitset out_bits

  the bits that flow out of the box
- vector< pair< AbstractBoxPtr, Connection > > dst_boxes

  describes how the `out_bits` flow from this box to other following boxes
- bool is_det

  a boolean value that should be true if and only if at least one out of all possible outputs has been determined and returned
- double prob

  the probability of the box to output the currently determined state

Friends

- class **RoundFunction**

4.2.1 Detailed Description

An AbstractBox represents an abstract idea of a block cipher component such as a Pbox, Sbox, Xor, Addition, etc. A cipher is composed of multiple such `boxes` that communicate with each other through connections.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AbstractBox() [1/2]

```
AbstractBox::AbstractBox (
            size_t in_size,
            size_t out_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes )
```

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
dst_boxes	output flow connections to following boxes

Precondition

for (auto &dst_box : dst_boxes) { dst_box.first != nullptr && dst_box.second.first.start + dst_box.second.↩ first.len <= out_bits.size() && dst_box.second.first.len == dst_box.second.second.len && dst_box.second.↩ second.start + dst_box.second.second.len <= dst_box.first->input_size(); }

4.2.2.2 AbstractBox() [2/2]

```
AbstractBox::AbstractBox (
            size_t in_size,
            size_t out_size )
```

similar to the previous constructor, but leaves `dst_boxes` empty

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box

Precondition

dst_box != nullptr && out_rng.start + out_rng.len <= out_bits.size() && in_rng.start + in_rng.len <= dst_box->in_bits.size();

4.2.3 Member Function Documentation

4.2.3.1 add_dest()

```
void AbstractBox::add_dest (
            AbstractBoxPtr dst_box,
            BitsRange out_range,
            BitsRange in_range )
```

add_dest adds a new destination box for the output of the box to flow to

Parameters

dst_box	a pointer to the destination box
out_range	a subrange of `out_bits` from this box that will flow to `dst_box`
in_range	a subrange of `in_bit` from `dst_box` into which the bits will flow

4.2.3.2 get_input()

```
const dynamic_bitset & AbstractBox::get_input ( )
```

getter for `in_bits`

Returns

`in_bits`

4.2.3.3 get_output()

```
const dynamic_bitset & AbstractBox::get_output ( )
```

getter for `out_bits`

Returns

`out_bits`

4.2.3.4 get_probability()

```
double AbstractBox::get_probability ( )
```

get the probability of the current characteristic

Returns

`prob`

4.2.3.5 input_size()

```
size_t AbstractBox::input_size ( )
```

getter for the size of `in_bits`

Returns

`in_bits.size()`

4.2.3.6 is_determined()

```
bool AbstractBox::is_determined ( )
```

getter for `is_det`

Returns

`is_det`

4.2.3.7 output_size()

```
size_t AbstractBox::output_size ( )
```

getter for the size of `out_bits`

Returns

`out_bits.size()`

4.2.3.8 set_input()

```
void AbstractBox::set_input (
            dynamic_bitset<> bits,
            const BitsRange & rng )  [virtual]
```

sets a subrange `rng` of the input to the value of `bits`

Parameters

bits	the bits that will be put in `in_bits`
rng	the subrange in which bits will be put in `in_bits`

Reimplemented in SBox.

The documentation for this class was generated from the following files:

- src/box/abstractbox.h
- src/box/abstractbox.cpp

4.3 BitsRange Struct Reference

BitsRange represents a way to represent a subrange of bits of a box.

`#include <helpers.h>`

Public Attributes

- size_t **start**
- size_t **len**

4.3.1 Detailed Description

BitsRange represents a way to represent a subrange of bits of a box.

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

- src/helpers/helpers.h

4.4 CipherAnalyzer Class Reference

Public Member Functions

- **CipherAnalyzer** (vector< RoundFunctionPtr > rounds, size_t input_max_hamming_weight, double global_thresh, vector< double > opt_probs)
- **CipherAnalyzer** (vector< RoundFunctionPtr > rounds, size_t input_max_hamming_weight, double global_thresh)
- ProbEntry **get_next_entry** ()
- void **set_input** (const dynamic_bitset<> &bits, BitsRange rng)

Protected Member Functions

- bool **advance_state** ()

Protected Attributes

- double **global_thresh**
- vector< double > **opt_probs**
- vector< double > **round_probs**
- vector< RoundFunctionPtr > **rounds**
- size_t **curr_idx**

The documentation for this class was generated from the following files:

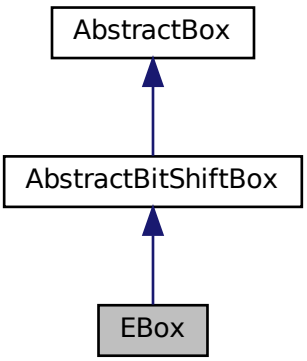
- src/cipheranalyzer.h
- src/cipheranalyzer.cpp

4.5 EBox Class Reference

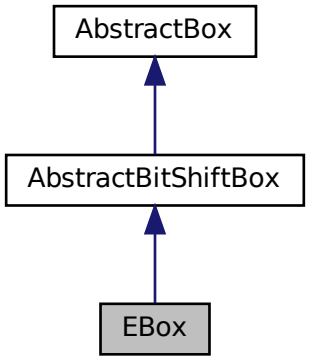
An EBox is a box that takes the input bits and expands them to the output by using some of multiple times.

`#include <ebox.h>`

Inheritance diagram for EBox:



Collaboration diagram for EBox:



Public Member Functions

- EBox (size_t in_size, size_t out_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes, const vector< size_t > &bit_expansion)
- EBox (size_t in_size, size_t out_size, const vector< size_t > &bit_expansion)

  similar to the previous constructor, but leaves `dst_boxes` empty

Additional Inherited Members

4.5.1 Detailed Description

An EBox is a box that takes the input bits and expands them to the output by using some of multiple times.

@inherits AbstractBitShiftBox

4.5.2 Constructor & Destructor Documentation

4.5.2.1 EBox() [1/2]

```
EBox::EBox (
            size_t in_size,
            size_t out_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes,
            const vector< size_t > & bit_expansion )
```

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
dst_boxes	output flow connections to following boxes
bit_expansion	an array used to compute to describe the expansion process of the input bits

See also

AbstractShiftBox constructor

Precondition

in_size <= out_size

4.5.2.2 EBox() [2/2]

```
EBox::EBox (
            size_t in_size,
            size_t out_size,
            const vector< size_t > & bit_expansion )
```

similar to the previous constructor, but leaves `dst_boxes` empty

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
bit_expansion	an array used to compute to describe the expansion process of the input bits

Precondition

in_size <= out_size

The documentation for this class was generated from the following files:

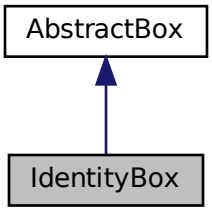
- src/box/ebox.h
- src/box/ebox.cpp

4.6 IdentityBox Class Reference

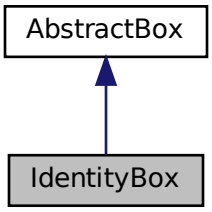
An IdentityBox is a box that represents the identity function `out_bits[i] = in_bits[i]`. An IdentityBox is useful as an accumulator for bits from multiple previous boxes which will then be sent to multiple following boxes.

`#include <identitybox.h>`

Inheritance diagram for IdentityBox:



Collaboration diagram for IdentityBox:



Public Member Functions

- IdentityBox (size_t data_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes)
- IdentityBox (size_t data_size)

  similar to the previous constructor, but leaves `dst_boxes` empty
- void determine_next () override

  since the identity function is a linear deterministic transformations, this will set `is_det` to `true` and `prob` to `1`

Additional Inherited Members

4.6.1 Detailed Description

An IdentityBox is a box that represents the identity function `out_bits[i] = in_bits[i]`. An IdentityBox is useful as an accumulator for bits from multiple previous boxes which will then be sent to multiple following boxes.

@inherits AbstractBox

4.6.2 Constructor & Destructor Documentation

4.6.2.1 IdentityBox() [1/2]

```
IdentityBox::IdentityBox (
            size_t data_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes )
```

Parameters

data_size	the size in bits of the input and the output
dst_boxes	output flow connections to following boxes

4.6.2.2 IdentityBox() [2/2]

```
IdentityBox::IdentityBox (
            size_t data_size )
```

similar to the previous constructor, but leaves `dst_boxes` empty

Parameters

data_size	the size in bits of the input and the output

The documentation for this class was generated from the following files:

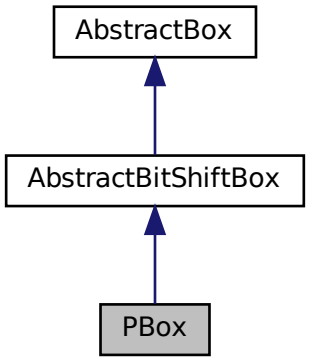
- src/box/identitybox.h
- src/box/identitybox.cpp

4.7 PBox Class Reference

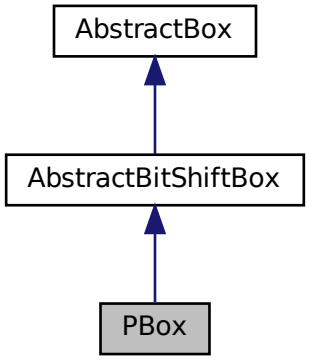
A PBox is a box that takes the input bits and permutes them to get the output.

```
#include <pbox.h>
```

Inheritance diagram for PBox:



Collaboration diagram for PBox:



Public Member Functions

- PBox (size_t data_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes, const vector< size_t > &bit_perm)
- PBox (size_t data_size, const vector< size_t > &bit_perm)

  PBox similar to the previous constructor, but leaves `dst_boxes` *empty.*

Additional Inherited Members

4.7.1 Detailed Description

A PBox is a box that takes the input bits and permutes them to get the output.

@inherits AbstractBitShiftBox

4.7.2 Constructor & Destructor Documentation

4.7.2.1 PBox() [1/2]

```
PBox::PBox (
            size_t data_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes,
            const vector< size_t > & bit_perm )
```

Parameters

data_size	the size in bits of the input and the output
dst_boxes	output flow connections to following boxes
bit_perm	a permutation used to shuffle the bits of the input to get the output

Precondition

bit_perm must be a permutation of [0, ..., bits_size - 1]

4.7.2.2 PBox() [2/2]

```
PBox::PBox (
            size_t data_size,
            const vector< size_t > & bit_perm )
```

PBox similar to the previous constructor, but leaves `dst_boxes` empty.

Parameters

data_size	the size in bits of the input and the output
bit_perm	a permutation used to shuffle the bits of the input to get the output

Precondition

bit_perm must be a permutation of [0, ..., bits_size - 1]

The documentation for this class was generated from the following files:

- src/box/pbox.h
- src/box/pbox.cpp

4.8 RoundFunction Class Reference

Public Member Functions

- **RoundFunction** (string src_id, string dst_id, map< string, AbstractBoxConstructor > constrs, map< string, vector< NamedConnection >> conns)
- bool **is_determined** ()
- ProbEntry **get_next_entry** ()
- void **set_input** (const dynamic_bitset<> bits, BitsRange rng)
- void **set_threshold** (double beta)

Protected Member Functions

- bool **advance_state** ()
- void **top_sort_boxes** (AbstractBoxPtr src, vector< AbstractBoxPtr > &top_sort, map< AbstractBoxPtr, bool > &is_visited)
- vector< AbstractBoxPtr > **sort_boxes** (AbstractBoxPtr src)

Protected Attributes

- AbstractBoxPtr **src**
- AbstractBoxPtr **dst**
- size_t **curr_box_idx**
- vector< AbstractBoxPtr > **boxes**
- vector< double > **partial_prob**
- double **beta_thresh**
- bool **is_det**

Friends

- class **CipherAnalyzer**

The documentation for this class was generated from the following files:

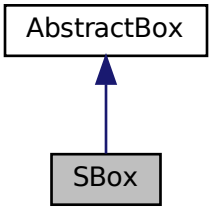
- src/roundfunction.h
- src/roundfunction.cpp

4.9 SBox Class Reference

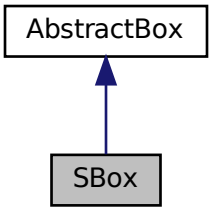
An SBox is a box that applies an arbitrary substitution on the input based on a substitution table. Since the actual value of the difference between pairs of inputs will change depending on the key bytes, this is a non-deterministic element in the cipher.

`#include <sbox.h>`

Inheritance diagram for SBox:



Collaboration diagram for SBox:



Public Member Functions

- SBox (size_t in_size, size_t out_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes, const ProbTable &prob_table)
- SBox (size_t in_size, size_t out_size, const ProbTable &prob_table)

  similar to the previous constructor, but leaves `dst_boxes` empty
- void determine_next () override

  determines the next best possible output for the given input, if all possible outputs have been determined, `is_det` is set to `true`
- void reset_determination () override

  sets `prob` to `0`, `is_det` to `false` and `table_idx` to `0`
- void set_input (dynamic_bitset<> bits, const BitsRange &rng) override

  sets a subrange `rng` of the input to the value of `bits`. Additionaly, `table_idx` to `0` and computes `table_entry` from `in_bits`

Protected Attributes

- ProbTable prob_table

  prob_table the probability table of the sbox
- size_t table_idx

  table_idx the current column in the row of the probability table. This is incremented each time `determine_next()` is called.
- size_t table_entry

  table_entry the row in the probability table corresponding to the value of `in_bits`

4.9.1 Detailed Description

An SBox is a box that applies an arbitrary substitution on the input based on a substitution table. Since the actual value of the difference between pairs of inputs will change depending on the key bytes, this is a non-deterministic element in the cipher.

@inherits AbstractBox

4.9.2 Constructor & Destructor Documentation

4.9.2.1 SBox() [1/2]

```
SBox::SBox (
            size_t in_size,
            size_t out_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes,
            const ProbTable & prob_table )
```

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
dst_boxes	output flow connections to following boxes
prob_table	the probability table of the sbox

4.9.2.2 SBox() [2/2]

```
SBox::SBox (
            size_t in_size,
            size_t out_size,
            const ProbTable & prob_table )
```

similar to the previous constructor, but leaves `dst_boxes` empty

Parameters

in_size	size of the input bits of this box
out_size	size of the output bits of this box
prob_table	the probability table of the sbox

4.9.3 Member Function Documentation

4.9.3.1 set_input()

```
void SBox::set_input (
            dynamic_bitset<> bits,
            const BitsRange & rng )  [override], [virtual]
```

sets a subrange `rng` of the input to the value of `bits`. Additionaly, `table_idx` to `0` and computes `table_entry` from `in_bits`

Parameters

bits	the bits that will be put in `in_bits`
rng	the subrange in which bits will be put in `in_bits`

Reimplemented from AbstractBox.

The documentation for this class was generated from the following files:

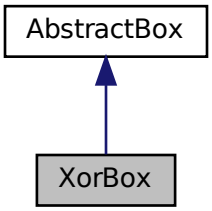
- src/box/sbox.h
- src/box/sbox.cpp

4.10 XorBox Class Reference

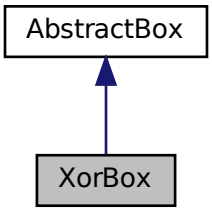
A XorBox is a box that computes the bitwise xor of two inputs. In order to simplify the implementation the the XorBox takes only one input `in_bits` but of size double of that of `out_bits`. the first half of `in_bits` represents the first of the two inputs, and the second half represents the last of the two inputs.

```
#include <xorbox.h>
```

Inheritance diagram for XorBox:



Collaboration diagram for XorBox:



Public Member Functions

- XorBox (size_t data_size, const vector< pair< AbstractBoxPtr, Connection >> &dst_boxes)
- XorBox (size_t data_size)

  similar to the previous constructor, but leaves `dst_boxes` *empty*
- void determine_next () override

  computes the xor between the first half and the second half of the `in_bits` *and stores it in* `out_bits`*. This is a linear and deterministic operation so* `is_det` *will be set to* `true` *and* `prob` *to* `1`

Additional Inherited Members

4.10.1 Detailed Description

A XorBox is a box that computes the bitwise xor of two inputs. In order to simplify the implementation the the XorBox takes only one input `in_bits` but of size double of that of `out_bits`. the first half of `in_bits` represents the first of the two inputs, and the second half represents the last of the two inputs.

@inherits AbstractBox

4.10.2 Constructor & Destructor Documentation

4.10.2.1 XorBox() [1/2]

```
XorBox::XorBox (
            size_t data_size,
            const vector< pair< AbstractBoxPtr, Connection >> & dst_boxes )
```

Parameters

data_size	the size in bits of the input and the output
dst_boxes	output flow connections to following boxes

4.10.2.2 XorBox() [2/2]

```
XorBox::XorBox (
            size_t data_size )
```

similar to the previous constructor, but leaves `dst_boxes` empty

Parameters

data_size	the size in bits of the input and the output

The documentation for this class was generated from the following files:

- src/box/xorbox.h
- src/box/xorbox.cpp

Chapter 5

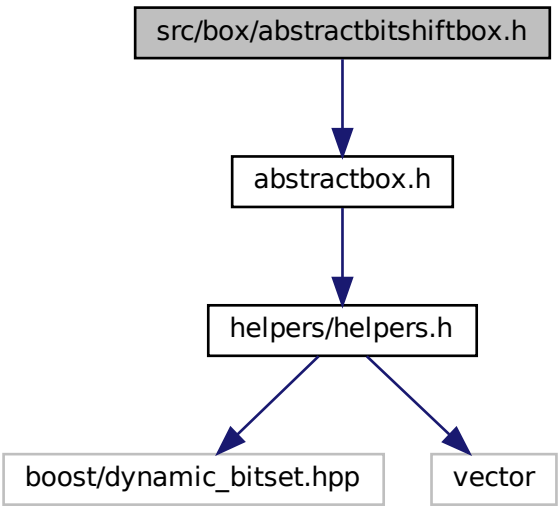
File Documentation

5.1 src/box/abstractbitshiftbox.h File Reference

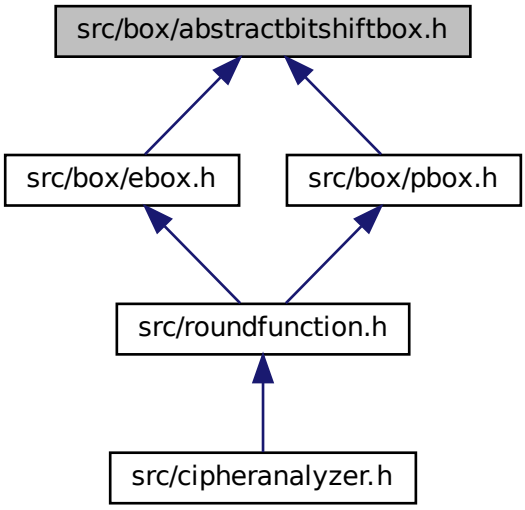
implementation of the AbstractBitShiftBox class

```
#include "abstractbox.h"
```

Include dependency graph for abstractbitshiftbox.h:



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



Classes

- class AbstractBitShiftBox

  An AbstractBitShiftBox represents a generic box that transforms input in output by shifting bits around.

Typedefs

- typedef std::shared_ptr< AbstractBitShiftBox > AbstractBitShiftBoxPtr

  shorthand for `std::shared_ptr<AbstractBitShiftBox>`

5.1.1 Detailed Description

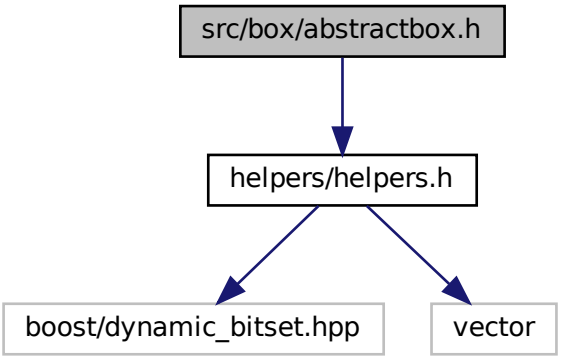
implementation of the AbstractBitShiftBox class

5.2 src/box/abstractbox.h File Reference

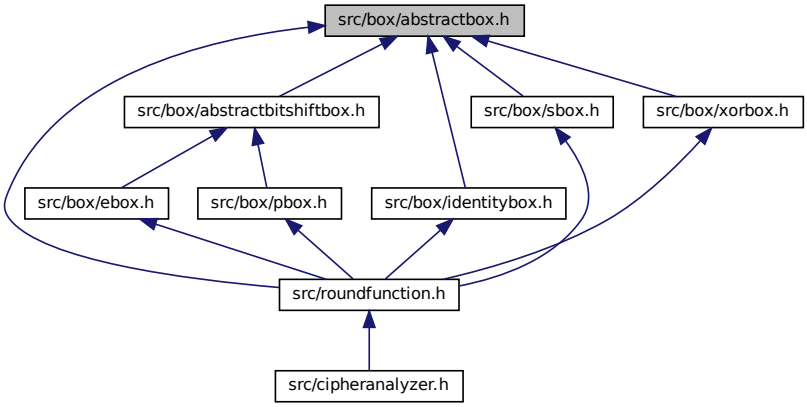
implementation of the AbstractBox class

```
#include "helpers/helpers.h"
```

Include dependency graph for abstractbox.h:



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



Classes

- class AbstractBox

  An AbstractBox represents an abstract idea of a block cipher component such as a Pbox, Sbox, Xor, Addition, etc. A cipher is composed of multiple such `boxes` that communicate with each other through connections.

Typedefs

- typedef std::shared_ptr< AbstractBox > AbstractBoxPtr

  shorthand for `std::shared_ptr<AbstractBox>`
- typedef function< AbstractBoxPtr() > AbstractBoxConstructor

  shorthand for `function<AbstractBoxPtr()>`. Represents a way to create an AbstractBoxPtr

5.2.1 Detailed Description

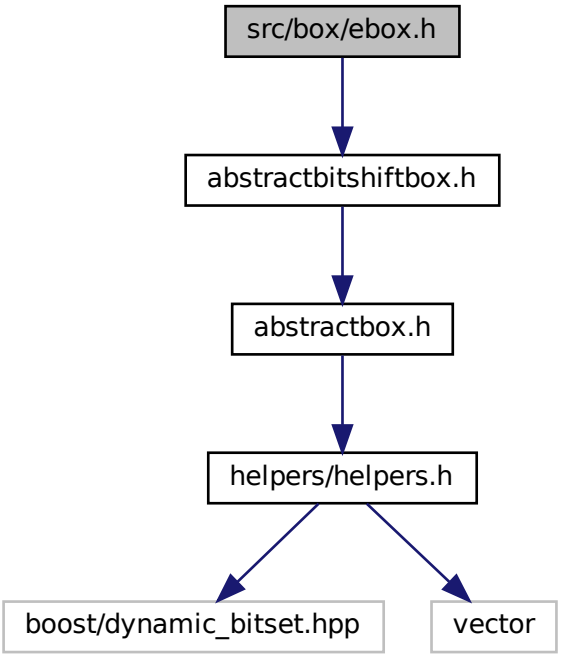
implementation of the AbstractBox class

5.3 src/box/ebox.h File Reference

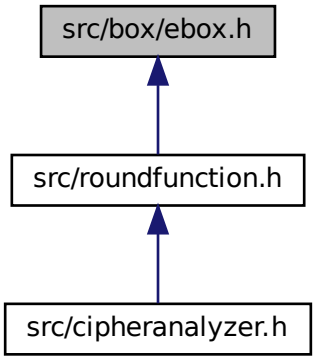
implementation of the EBox class

```
#include "abstractbitshiftbox.h"
```

Include dependency graph for ebox.h:



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



Classes

- class EBox

  An EBox is a box that takes the input bits and expands them to the output by using some of multiple times.

Typedefs

- typedef std::shared_ptr< EBox > EBoxPtr

  shorthand for std::shared_ptr<Ebox>

5.3.1 Detailed Description

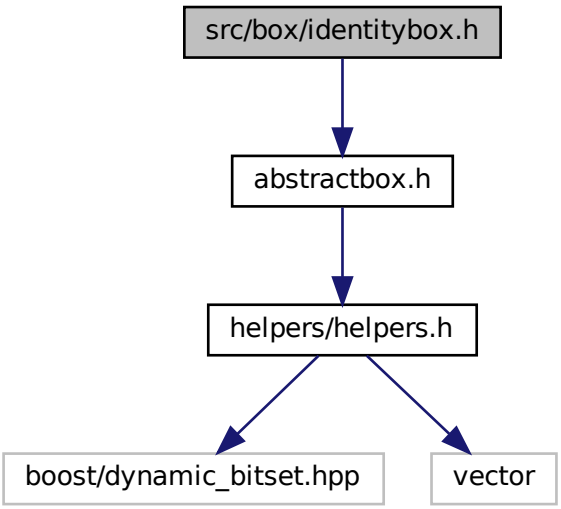
implementation of the EBox class

5.4 src/box/identitybox.h File Reference

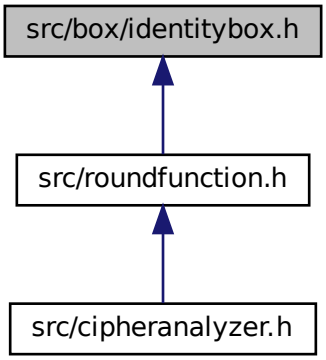
implementation of the IdentityBox class

```
#include "abstractbox.h"
```

Include dependency graph for identitybox.h:



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



Classes

- class IdentityBox

  An IdentityBox is a box that represents the identity function `out_bits[i] = in_bits[i]`. An IdentityBox is useful as an accumulator for bits from multiple previous boxes which will then be sent to multiple following boxes.

Typedefs

- typedef std::shared_ptr< IdentityBox > IdentityBoxPtr

  shorthand for std::shared_ptr<IdentityBox>

5.4.1 Detailed Description

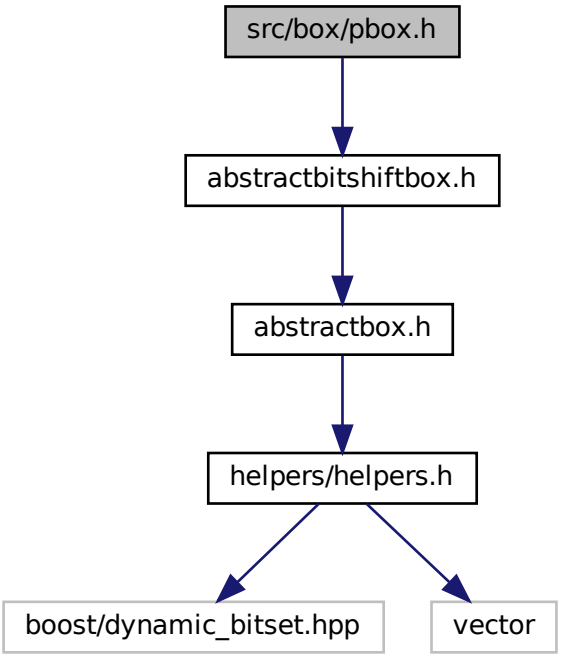
implementation of the IdentityBox class

5.5 src/box/pbox.h File Reference

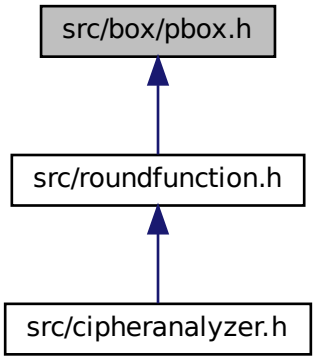
implementation of the PBox class

```
#include "abstractbitshiftbox.h"
```

Include dependency graph for pbox.h:



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



Classes

- class PBox

  A PBox is a box that takes the input bits and permutes them to get the output.

Typedefs

- typedef std::shared_ptr< PBox > PBoxPtr

  shorthand for std::shared_ptr<PBox>

5.5.1 Detailed Description

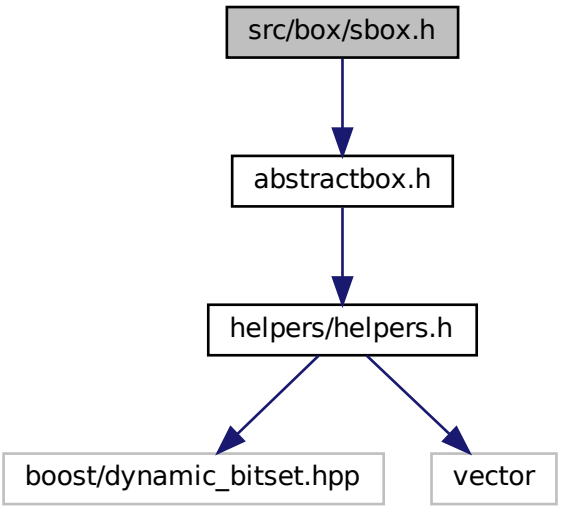
implementation of the PBox class

5.6 src/box/sbox.h File Reference

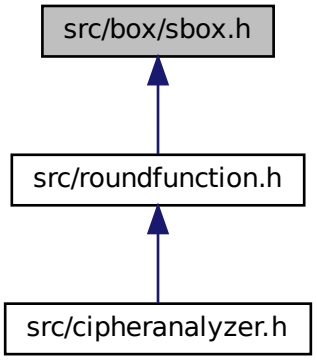
implementation of the SBox class

```
#include "abstractbox.h"
```

Include dependency graph for sbox.h:



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



Classes

- class SBox

  An SBox is a box that applies an arbitrary substitution on the input based on a substitution table. Since the actual value of the difference between pairs of inputs will change depending on the key bytes, this is a non-deterministic element in the cipher.

Typedefs

- typedef std::shared_ptr< SBox > SBoxPtr

  shorthand for std::shared_ptr<SBox>

5.6.1 Detailed Description

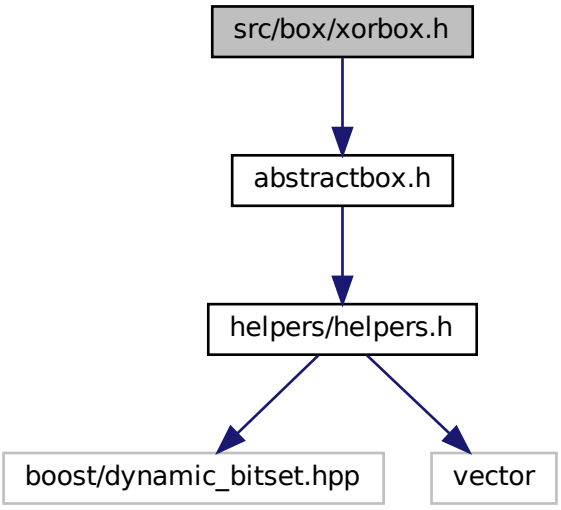
implementation of the SBox class

5.7 src/box/xorbox.h File Reference

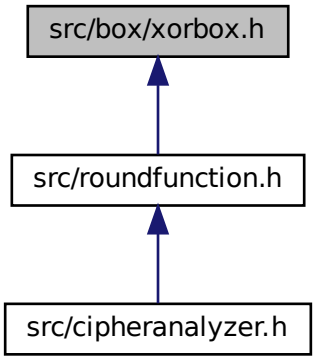
implementation of the XorBox class

```
#include "abstractbox.h"
```

Include dependency graph for xorbox.h:



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



Classes

- class XorBox

  A XorBox is a box that computes the bitwise xor of two inputs. In order to simplify the implementation the the XorBox takes only one input `in_bits` but of size double of that of `out_bits`. the first half of `in_bits` represents the first of the two inputs, and the second half represents the last of the two inputs.

Typedefs

- typedef std::shared_ptr< XorBox > XorBoxPtr

  shorthand for std::shared_ptr<XorBox>

5.7.1 Detailed Description

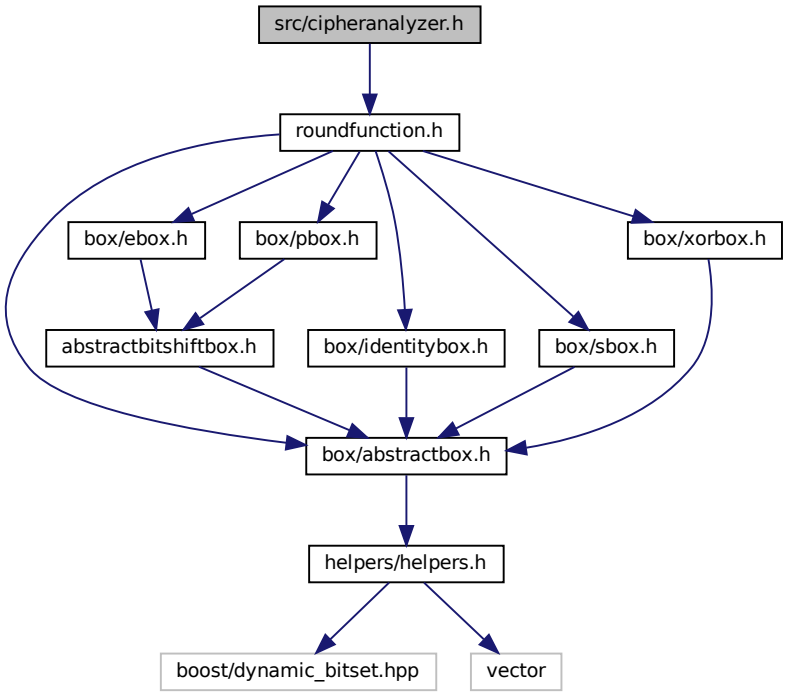
implementation of the XorBox class

5.8 src/cipheranalyzer.h File Reference

implementation of the CipherAnalyzer class

```
#include "roundfunction.h"
```

Include dependency graph for cipheranalyzer.h:



Classes

- class CipherAnalyzer

Typedefs

- typedef std::shared_ptr< CipherAnalyzer > **CipherAnalyzerPtr**

5.8.1 Detailed Description

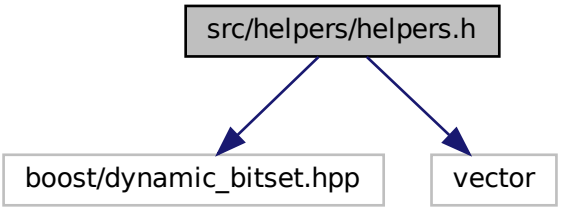
implementation of the CipherAnalyzer class

5.9 src/helpers/helpers.h File Reference

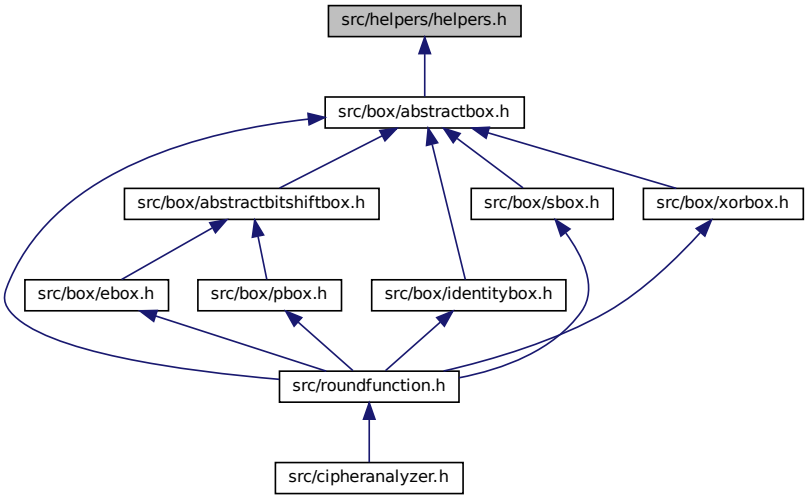
A collection of helper functions and structs.

```
#include <boost/dynamic_bitset.hpp>
#include <vector>
```

Include dependency graph for helpers.h:



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



Classes

- struct BitsRange

  BitsRange represents a way to represent a subrange of bits of a box.

Typedefs

- typedef pair< BitsRange, BitsRange > Connection

  shorthand for pair<BitsRange, BitsRange>. Represents a connection between two boxes, the first BitsRange is the subrange of the first box, while the second BitsRange is the subrange of the second box
- typedef pair< string, Connection > NamedConnection

shorthand for pair<string, Connection>

- typedef pair< dynamic_bitset<>, double > ProbEntry

  shorthand for pair<dynamic_bitset<>, double>. Represents a pair between an output and it's probability of an sbox given it's input

- typedef vector< ProbEntry > ProbTableLine

  shorthand for vector<ProbEntry>. Represents a whole line of a probability table

- typedef vector< ProbTableLine > ProbTable

  shorthand for vector<ProbTableLine>. A probability table is a compact way to describe the behaviour of an sbox when analyzing the difference of two inputs.

Functions

- ProbTable compute_diff_dist_table (const vector< size_t > &sbox)

  computes the difference distribution table of an sbox. Used for differential cryptanalysis

- bool increment_bits (dynamic_bitset<> &input)

  given some input, calculates the next lexicografically smallest binary array with the same number of bits. It sorts all entries for an input by their probability

- dynamic_bitset to_dynamic_bitset (size_t input, size_t bit_size)

  converts a `size_t` to a `dynamic_bitset` of size `bit_size`

- dynamic_bitset to_dynamic_bitset (unsigned int input, size_t bit_size)

  converts an `unsigned int` to a `dynamic_bitset` of size `bit_size`

- size_t convert_to_index (const dynamic_bitset<> &bits)

  converts a `dynamic_bitset` to a `size_t`

- unsigned int convert_to_uint (const dynamic_bitset<> &bits)

  converts a `dynamic_bitset` to an `unsigned int`

5.9.1 Detailed Description

A collection of helper functions and structs.

5.9.2 Function Documentation

5.9.2.1 compute_diff_dist_table()

```
ProbTable compute_diff_dist_table (
            const vector< size_t > & sbox )
```

computes the difference distribution table of an sbox. Used for differential cryptanalysis

Parameters

sbox	a substitution function

Returns

the difference distribution table (DDT) of the sbox

Precondition

input_size (i.e. sbox.size()) and output_size (i.e. max(sbox) + 1) should be powers of two

5.9.2.2 convert_to_index()

```
size_t convert_to_index (
            const dynamic_bitset<> & bits )
```

converts a `dynamic_bitset` to a `size_t`

Parameters

bits	a `dynamic_bitset` representing a binary array

Returns

a `size_t` representing the number formed from the binray array given as input

5.9.2.3 convert_to_uint()

```
unsigned int convert_to_uint (
            const dynamic_bitset<> & bits )
```

converts a `dynamic_bitset` to an `unsigned int`

Parameters

bits	a `dynamic_bitset` representing a binary array

Returns

an `unsigned int` representing the number formed from the binray array given as input

5.9.2.4 increment_bits()

```
bool increment_bits (
            dynamic_bitset<> & input )
```

given some input, calculates the next lexicografically smallest binary array with the same number of bits. It sorts all entries for an input by their probability

Parameters

input	a binary array

Returns

`true` if there was at least one next

5.9.2.5 to_dynamic_bitset() [1/2]

```
dynamic_bitset to_dynamic_bitset (
            size_t input,
            size_t bit_size )
```

converts a `size_t` to a `dynamic_bitset` of size `bit_size`

Parameters

input	a `size_t`
bit_size	the size in bits

Returns

a `dynamic_bitset<>` representing the binary array computed from input

5.9.2.6 to_dynamic_bitset() [2/2]

```
dynamic_bitset to_dynamic_bitset (
            unsigned int input,
            size_t bit_size )
```

converts an `unsigned int` to a `dynamic_bitset` of size `bit_size`

Parameters

input	a `size_t`
bit_size	the size in bits

Returns

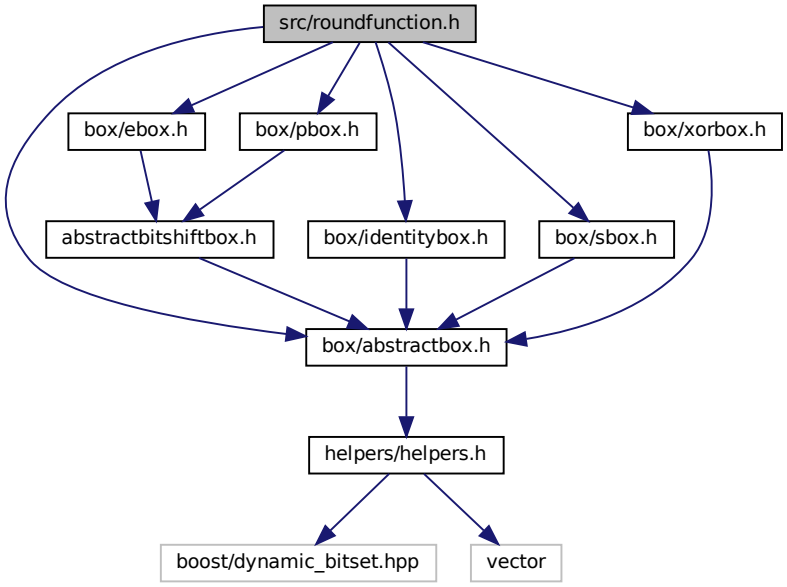
a `dynamic_bitset<>` representing the binary array computed from input

5.10 src/roundfunction.h File Reference

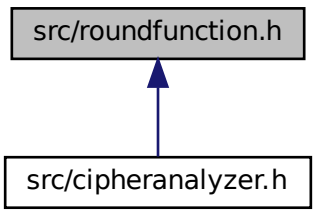
implementation of the RoundFunction class

```
#include "box/abstractbox.h"
#include "box/ebox.h"
#include "box/identitybox.h"
#include "box/pbox.h"
#include "box/sbox.h"
#include "box/xorbox.h"
```

Include dependency graph for roundfunction.h:



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



Classes

- class RoundFunction

Typedefs

- typedef std::shared_ptr< RoundFunction > **RoundFunctionPtr**

5.10.1 Detailed Description

implementation of the RoundFunction class

Index