Glite Security encrypted storage cpp Reference Manual

Version 0.8.3

Generated by Doxygen 1.3.5

Thu May 11 18:54:33 2006

Contents

1 Cryptographic Tools for glite EDS 1

2 Glite Security encrypted storage cpp Namespace Index 1

3 Glite Security encrypted storage cpp Class Index 2

4 Glite Security encrypted storage cpp File Index 2

5 Glite Security encrypted storage cpp Namespace Documentation 2

6 Glite Security encrypted storage cpp Class Documentation 7

7 Glite Security encrypted storage cpp File Documentation 23

1 Cryptographic Tools for glite EDS

1.1 Introduction

This is the documentation of the security module that contain cryptographic tools to be used in the gLite grid middleware.

1.2 Conditions

Author:
Patrick Guio <patrick.guio@bccs.uib.no>

2 Glite Security encrypted storage cpp Namespace Index

2.1 Glite Security encrypted storage cpp Namespace List

Here is a list of all namespaces with brief descriptions:

glite 3

3 Glite Security encrypted storage cpp Class Index

3.1 Glite Security encrypted storage cpp Class List

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

glite::assignXor< Tp > 7

glite::bitwiseXor< Tp > 8

glite::Crypt (Class for encryted data storage) 8

glite::inputKey< key_type > (State structure manipulator to input key) 19

glite::outputKey< key_type > (State structure manipulator to output key) 20

glite::setFormat (State structure manipulator to format key) 22

4 Glite Security encrypted storage cpp File Index

4.1 Glite Security encrypted storage cpp File List

Here is a list of all files with brief descriptions:

src/**base64.cc** (Definitions of member functions for Base64 encoding/decoding) 23

src/**crypt.cc** (Definitions of member functions for Pseudo-random key and initialisation vector generation Encryption and decryption blocks and files Miscelleanous modifiers and queries Error handling) 23

src/**crypt.h** (Declarations for gLite encryted data storage) 23

src/**io.cc** (Definitions for key I/O manipulation) 24

src/**shamir.cc** (Definition for Shamir secret sharing (SSS) scheme) 25

src/**test-base64.cc** 25

src/**test-shamir.cc** 26

src/**test-tss.cc** 26

src/**tss.cc** (Definitions for Trivial secret sharing (TSS) scheme) 27

5 Glite Security encrypted storage cpp Namespace Documentation

5.1 glite Namespace Reference

Classes

- struct assignXor
- struct bitwiseXor
- class Crypt

  Class for encryted data storage.

- struct inputKey

  State structure manipulator to input key.

- struct outputKey

  State structure manipulator to output key.

- struct setFormat

  State structure manipulator to format key.

Functions

- template<typename key_type> key_type Array2Key (const typename key_type::value_type *array, unsigned int length)

  Convert an array into a key.

- template Crypt::Key Array2Key (const Crypt::Key::value_type *array, unsigned int length)
- template Crypt::Base64Key Array2Key (const Crypt::Base64Key::value_type *array, unsigned int length)
- template Crypt::ShortKey Array2Key (const Crypt::ShortKey::value_type *array, unsigned int length)
- template<typename key_type> key_type::value_type * Key2Array (const key_type &vec, unsigned int &length)

  Convert a key into an array.

- template Crypt::Key::value_type * Key2Array (const Crypt::Key &vec, unsigned int &length)
- template Crypt::Base64Key::value_type * Key2Array (const Crypt::Base64Key &vec, unsigned int &length)
- template Crypt::ShortKey::value_type * Key2Array (const Crypt::ShortKey &vec, unsigned int &length)
- std::ostream & operator<< (std::ostream &os, const Crypt &c)

  Output insertion operator for object of class Crypt.

- template<typename key_type> std::ostream & operator<< (std::ostream &os, const outputKey< std::vector< key_type > > &k)

  Output insertion operator for object of type key.

- std::ostream & operator<< (std::ostream &os, const outputKey< Crypt::Base64Key > &k)

  Output insertion operator for object of type Base64 key.

- template<typename key_type> std::ostream & operator<< (std::ostream &os, const outputKey< std::vector< std::vector< key_type > > > &k)

Output insertion operator for object of type split keys.

- template<typename key_type> std::istream & operator>> (std::istream &is, inputKey< key_type > &k)

  Input insertion operator for object of type key.

- std::istream & operator>> (std::istream &is, const setFormat &f)

  Input insertion operator for object of type format.

- std::ostream & operator<< (std::ostream &os, const setFormat &f)

  Output insertion operator for object of type format.

- template std::ostream & operator<< (std::ostream &os, const outputKey< Crypt::Key > &k)
- template std::ostream & operator<< (std::ostream &os, const outputKey< Crypt::ShortKey > &k)
- template std::ostream & operator<< (std::ostream &os, const outputKey< Crypt::LongKey > &k)
- template std::ostream & operator<< (std::ostream &os, const outputKey< Crypt::SplitKeys > &k)
- template std::ostream & operator<< (std::ostream &os, const outputKey< Crypt::SplitShortKeys > &k)
- template std::istream & operator>> (std::istream &is, inputKey< Crypt::Key > &k)
- template std::istream & operator>> (std::istream &is, inputKey< Crypt::ShortKey > &k)
- template std::istream & operator>> (std::istream &is, inputKey< Crypt::LongKey > &k)
- template<typename InputIterator1, typename InputIterator2, typename BinaryOperator> InputIterator2 transform (InputIterator1 first1, InputIterator1 last1, InputIterator2 first2, BinaryOperator binary_op)

Variables

- const int evp_max_block_length = 32
- const Crypt::longByte prime = 65521

  Value of the prime number for modular arithmetic.

5.1.1 Function Documentation

5.1.1.1 template Crypt::ShortKey Array2Key (const Crypt::ShortKey::value_type * *array*, unsigned int *length*)

5.1.1.2 template Crypt::Base64Key Array2Key (const Crypt::Base64Key::value_type * *array*, unsigned int *length*)

5.1.1.3 template Crypt::Key Array2Key (const Crypt::Key::value_type * *array*, unsigned int *length*)

5.1.1.4 template<typename key_type> key_type glite::Array2Key (const typename key_type::value_type * *array*, unsigned int *length*)

Convert an array into a key.

Parameters:

in] array pointer to an array block

in] length array size

Returns:

key converted from array

5.1.1.5 template Crypt::ShortKey::value_type* Key2Array (const Crypt::ShortKey & *vec*, unsigned int & *length*)

5.1.1.6 template Crypt::Base64Key::value_type* Key2Array (const Crypt::Base64Key & *vec*, unsigned int & *length*)

5.1.1.7 template Crypt::Key::value_type* Key2Array (const Crypt::Key & *vec*, unsigned int & *length*)

5.1.1.8 template<typename key_type> key_type::value_type * glite::Key2Array (const key_type & *vec*, unsigned int & *length*)

Convert a key into an array.

Parameters:

in] key

out] length array size

Returns:

pointer to an array block (allocated with new, freed with delete)

5.1.1.9 template std::ostream& operator<< (std::ostream & *os*, const outputKey< Crypt::SplitShortKeys > & *k*)

5.1.1.10 template std::ostream& operator<< (std::ostream & *os*, const outputKey< Crypt::SplitKeys > & *k*)

5.1.1.11 template std::ostream& operator<< (std::ostream & *os*, const outputKey< Crypt::LongKey > & *k*)

5.1.1.12 template std::ostream& operator<< (std::ostream & *os*, const outputKey< Crypt::ShortKey > & *k*)

5.1.1.13 template std::ostream& operator<< (std::ostream & *os*, const outputKey< Crypt::Key > & *k*)

5.1.1.14 std::ostream & glite::operator<< (std::ostream & *os*, const setFormat & *f*)

Output insertion operator for object of type format.

5.1.1.15 template<typename key_type> std::ostream & glite::operator<< (std::ostream & *os*, const outputKey< std::vector< std::vector< key_type > > > & *k*)

Output insertion operator for object of type split keys.

5.1.1.16 std::ostream & glite::operator<< (std::ostream & *os*, const outputKey< Crypt::Base64Key > & *k*)

Output insertion operator for object of type Base64 key.

5.1.1.17 template<typename key_type> std::ostream & glite::operator<< (std::ostream & *os*, const outputKey< std::vector< key_type > > & *k*)

Output insertion operator for object of type key.

5.1.1.18 std::ostream & glite::operator<< (std::ostream & *os*, const Crypt & *c*)

Output insertion operator for object of class Crypt.

5.1.1.19 template std::istream& operator>> (std::istream & *is*, inputKey< Crypt::LongKey > & *k*)

5.1.1.20 template std::istream& operator>> (std::istream & *is*, inputKey< Crypt::ShortKey > & *k*)

5.1.1.21 template std::istream& operator>> (std::istream & *is*, inputKey< Crypt::Key > & *k*)

5.1.1.22 std::istream & glite::operator>> (std::istream & *is*, const setFormat & *f*)

Input insertion operator for object of type format.

5.1.1.23 template<typename key_type> std::istream & glite::operator>> (std::istream & *is*, inputKey< key_type > & *k*)

Input insertion operator for object of type key.

5.1.1.24 template<typename InputIterator1, typename InputIterator2, typename BinaryOperator> InputIterator2 transform (InputIterator1 *first1*, InputIterator1 *last1*, InputIterator2 *first2*, BinaryOperator *binary_op*)

5.1.2 Variable Documentation

5.1.2.1 const int glite::evp_max_block_length = 32 [static]

5.1.2.2 const Crypt::longByte glite::prime = 65521 [static]

Value of the prime number for modular arithmetic.

6 Glite Security encrypted storage cpp Class Documentation

6.1 glite::assignXor< Tp > Struct Template Reference

template<class Tp> struct glite::assignXor< Tp >

Public Member Functions

- void operator() (Tp &x, const Tp &y) const

6.1.1 Member Function Documentation

6.1.1.1 template<class Tp> void glite::assignXor< Tp >::operator() (Tp & *x*, const Tp & *y*) const [inline]

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

- src/tss.cc

6.2 glite::bitwiseXor< Tp > Struct Template Reference

template<class Tp> struct glite::bitwiseXor< Tp >

Public Member Functions

- Tp operator() (const Tp &x, const Tp &y) const

6.2.1 Member Function Documentation

6.2.1.1 template<class Tp> Tp glite::bitwiseXor< Tp >::operator() (const Tp & *x*, const Tp & *y*) const [inline]

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

- src/tss.cc

6.3 glite::Crypt Class Reference

`#include <crypt.h>`

6.3.1 Detailed Description

Class for encryted data storage.

Available methods to

- encrypt/decrypt a buffer or a file using an encryption key and an algorithm implementation from OpenSSL.
- Base64 encode/decode a key.
- split/join key using Trivial Sharing Secret scheme.
- split/join key using Shamir Sharing Secret scheme.

Public Types

- typedef unsigned char byte
  default key type
- typedef std::vector< byte > Key
  container for key
- typedef std::string Base64Key
  container for Base64 key.
- typedef std::vector< Key > SplitKeys
  container for TSS split keys.
- typedef unsigned short shortByte
  key type for SSS key
- typedef std::vector< shortByte > ShortKey
  container for SSS key
- typedef std::vector< shortByte > Polynom
  container for SSS polynom representation.
- typedef std::vector< ShortKey > SplitShortKeys
  container for SSS split keys.
- typedef unsigned long longByte
  key type for SSS key processing
- typedef std::vector< longByte > LongKey
  container for SSS key processing

Public Member Functions

- Crypt (const std::string cipher="bf-cbc", int keyLength=0)
  default constructor
- ~Crypt ()
  default destructor
- void encrypt (byte *in, int isize, byte *&out, int &osize) const
  encrypts a data buffer
- void encrypt (const std::string &ifile, const std::string &ofile) const
  encrypts a file
- void decrypt (byte *in, int isize, byte *&out, int &osize) const
  decrypts an input data buffer

- void decrypt (const std::string &ifile, const std::string &ofile) const

  decrypts a file

- void encodeBase64 (const Key &key, Base64Key &b64key) const

  Base64 encode an input key.

- void decodeBase64 (const Base64Key &b64key, Key &key) const

  Base64 decode an input key.

- void splitKeyTSS (const Key &key, int nShares, SplitKeys &keys) const

  Split Key using Trivial Secret Sharing.

- void joinKeyTSS (const SplitKeys &keys, Key &key) const

  Join Keys using Trivial Secret Sharing.

- void splitKeySSS (const Key &key, int nNeeded, int nShares, Key &x, SplitShortKeys &keys) const

  Split Key using Shamir Secret Sharing.

- void joinKeySSS (const int nNeeded, const Key &x, const SplitShortKeys &keys, Key &key) const

  Join Keys using Shamir Secret Sharing.

- void setKeyAndIV (const Key &key, const Key &iv=Key())

  Set an encryption/decryption key and initialisation vector (optional).

- Key getKey () const

  Return current encryption/decryption key.

- Key getIV () const

  Return current initialisation vector.

- void setBufferSize (int size)

  Set buffer size to encrypt/decrypt file.

- int getBufferSize () const

  Return current buffer size.

- void setVerbose (int verbose)

  Set verbose level.

- int getVerbose () const

  Return current verbose level.

- void printOn (std::ostream &os) const

  Print state of Crypter object.

Private Member Functions

- void seedPRNG (int bytes=1) const

  Seed the pseudo random number generator.

- void drawKey ()

  Draw a pseudo random key and set the encryption/decryption key.

- void drawIV ()

  Draw a pseudo-random initialisation vector and set the initialisation vector.

- void initCrypters ()

  Initialise Crypter object for encryption/decryption.

- void handleError (const char *thisFile, int thisLine) const

  Handle error and throw an exception.

- void drawPRN (Key &key) const

  Draw a random key (TSS).

- void drawPRN (Polynom &p) const

  Draw a random polynom (SSS).

- Polynom initPolynom (int k, byte a0) const

  Initialise a polynomial to be used in SSS.

- longByte evalPolynom (const Polynom &p, const longByte x) const

  Evaluate a polynomial at a given value (SSS).

- longByte inverseModulo (const longByte n) const

  Calculate the modular (prime) inverse of a number.

- longByte divideModulo (const longByte numerator, const longByte denominator) const

  Compute the modular (prime) division of two numbers.

- void evalLagrangeInterpAt0 (const Key &x, LongKey &numerator, LongKey &denominator) const

  Compute Lagrange interpolation at zero.

Private Attributes

- const std::string m_cipherName

  name of OpenSSL cypher type

- unsigned int m_keyLength

  encryption/decryption key length

- unsigned int m_ivLength

  initialisation vector length

- int m_bufferSize
  buffer size to encrypt/decrypt file
- EVP_CIPHER_CTX * m_ectx
  OpenSSL encrypting cipher context.
- EVP_CIPHER_CTX * m_dctx
  OpenSSL decrypting cipher context.
- const EVP_CIPHER * m_type
  OpenSSL EVP_CIPHER structure.
- Key m_key
  encryption/decryption key
- Key m_iv
  initialisation vector
- int m_verbose
  verbose level

6.3.2 Member Typedef Documentation

6.3.2.1 typedef std::string glite::Crypt::Base64Key

container for Base64 key.

6.3.2.2 typedef unsigned char glite::Crypt::byte

default key type

6.3.2.3 typedef std::vector<byte> glite::Crypt::Key

container for key

6.3.2.4 typedef unsigned long glite::Crypt::longByte

key type for SSS key processing

6.3.2.5 typedef std::vector<longByte> glite::Crypt::LongKey

container for SSS key processing

6.3.2.6 typedef std::vector<shortByte> glite::Crypt::Polynom

container for SSS polynom representation.

6.3.2.7 typedef unsigned short glite::Crypt::shortByte

key type for SSS key

6.3.2.8 typedef std::vector<shortByte> glite::Crypt::ShortKey

container for SSS key

6.3.2.9 typedef std::vector<Key> glite::Crypt::SplitKeys

container for TSS split keys.

6.3.2.10 typedef std::vector<ShortKey > glite::Crypt::SplitShortKeys

container for SSS split keys.

6.3.3 Constructor & Destructor Documentation

6.3.3.1 glite::Crypt::Crypt (const std::string *cipherName* = "bf-cbc", int *keyLength* = 0)

default constructor

builds an encrypter/decrypter and initialise the associated OpenSSL context

Parameters:

in] cipherName OpenSSL cipher name (see man enc for a list of supported ciphers)

in] keyLength length of the encryption key in bits

6.3.3.2 glite::Crypt::~Crypt ()

default destructor

6.3.4 Member Function Documentation

6.3.4.1 void glite::Crypt::decodeBase64 (const Base64Key & *b64message*, Key & *message*) const

Base64 decode an input key.

Parameters:

in] b64message input Base64 key to decode

out] message output decoded key

6.3.4.2 void glite::Crypt::decrypt (const std::string & *ifile*, const std::string & *ofile*) const

decrypts a file

Parameters:

in] ifile input filename.

in] ofile output filename.

6.3.4.3 void glite::Crypt::decrypt (byte * *in*, int *isize*, byte *& *out*, int & *osize*) const

decrypts an input data buffer

Parameters:

in] in input data buffer

in] isize input data buffer size

out] out output data buffer Newly allocated decrypted data buffer: it is up to the caller to deallocate it

out] osize returned size of decrypted data buffer

6.3.4.4 Crypt::longByte glite::Crypt::divideModulo (const longByte *numerator*, const longByte *denominator*) const [private]

Compute the modular (prime) division of two numbers.

6.3.4.5 void glite::Crypt::drawIV () [private]

Draw a pseudo-random initialisation vector and set the initialisation vector.

6.3.4.6 void glite::Crypt::drawKey () [private]

Draw a pseudo random key and set the encryption/decryption key.

6.3.4.7 void glite::Crypt::drawPRN (Polynom & *p*) const [private]

Draw a random polynom (SSS).

6.3.4.8 void glite::Crypt::drawPRN (Key & *key*) const [private]

Draw a random key (TSS).

6.3.4.9 void glite::Crypt::encodeBase64 (const Key & *message*, Base64Key & *b64message*) const

Base64 encode an input key.

Parameters:

in] message input key to encode

out] b64message output encoded key

6.3.4.10 void glite::Crypt::encrypt (const std::string & *ifile*, const std::string & *ofile*) const

encrypts a file

Parameters:

in] ifile input filename.

in] ofile output filename.

6.3.4.11 void glite::Crypt::encrypt (byte * *in*, int *isize*, byte *& *out*, int & *osize*) const

encrypts a data buffer

Parameters:

in] in input data buffer.

in] isize input data buffer size.

out] out output data buffer. Newly allocated encrypted data buffer: it is up to the caller to deallocate it.

out] osize returned size of encrypting data buffer (must be known for decrypting)

6.3.4.12 void glite::Crypt::evalLagrangeInterpAt0 (const Key & *x*, LongKey & *numerator*, LongKey & *denominator*) const [private]

Compute Lagrange interpolation at zero.

6.3.4.13 Crypt::longByte glite::Crypt::evalPolynom (const Polynom & *p*, const longByte *x*) const [private]

Evaluate a polynomial at a given value (SSS).

6.3.4.14 int glite::Crypt::getBufferSize () const

Return current buffer size.

6.3.4.15 Crypt::Key glite::Crypt::getIV () const

Return current initialisation vector.

6.3.4.16 Crypt::Key glite::Crypt::getKey () const

Return current encryption/decryption key.

6.3.4.17 int glite::Crypt::getVerbose () const

Return current verbose level.

6.3.4.18 void glite::Crypt::handleError (const char * *thisFile*, int *thisLine*) const [private]

Handle error and throw an exception.

6.3.4.19 void glite::Crypt::initCrypters () [private]

Initialise Crypter object for encryption/decryption.

6.3.4.20 Crypt::Polynom glite::Crypt::initPolynom (int *k*, byte *a0*) const [private]

Initialise a polynomial to be used in SSS.

Polynomial represented in descending powers

$$Y = P(0) * X^{k-1} + P(1) * X^{k-2} + ... + P(k-2) * X + P(k-1)$$

where $P(k-1)$ is the secret to share and all other coefficients are randomly chosen numbers modulo prime.

6.3.4.21 Crypt::longByte glite::Crypt::inverseModulo (const longByte *n*) const [private]

Calculate the modular (prime) inverse of a number.

calculate the modular inverse using Extended Euclidean Algorithm

The Extended Euclidean algorithm not only computes gcd(n,m), but also returns the numbers a and b such that gcd(n,m)=a∗n+b∗m. If gcd(n,m)=1 this solves the problem of computing modular inverses, which is the case for gcd(prime,n) = 1

6.3.4.22 void glite::Crypt::joinKeySSS (const int *nNeeded*, const Key & *x*, const SplitShortKeys & *keys*, Key & *key*) const

Join Keys using Shamir Secret Sharing.

Parameters:

in] nNeeded keys to join

in] x values at interpolation

in] keys to join

out] key joint key

6.3.4.23 void glite::Crypt::joinKeyTSS (const SplitKeys & *keys*, Key & *key*) const

Join Keys using Trivial Secret Sharing.

Parameters:

in] keys input keys to join

out] key output join key

6.3.4.24 void glite::Crypt::printOn (std::ostream & *os*) const

Print state of Crypter object.

6.3.4.25 void glite::Crypt::seedPRNG (int *bytes* = 1) const [private]

Seed the pseudo random number generator.

6.3.4.26 void glite::Crypt::setBufferSize (int *size*)

Set buffer size to encrypt/decrypt file.

6.3.4.27 void glite::Crypt::setKeyAndIV (const Key & *key*, const Key & *iv* = Key())

Set an encryption/decryption key and initialisation vector (optional).

6.3.4.28 void glite::Crypt::setVerbose (int *verbose*)

Set verbose level.

6.3.4.29 void glite::Crypt::splitKeySSS (const Key & *key*, int *nNeeded*, int *nShares*, Key & *x*, SplitShortKeys & *keys*) const

Split Key using Shamir Secret Sharing.

Parameters:

in] key input key to split

in] nNeeded input number of shares needed

in] nShares input number of shares

out] x output

out] keys output split keys

6.3.4.30 void glite::Crypt::splitKeyTSS (const Key & *key*, int *n*, SplitKeys & *keys*) const

Split Key using Trivial Secret Sharing.

Parameters:

in] key input key to split

in] nShares input number of shares

out] keys output split keys

6.3.5 Member Data Documentation

6.3.5.1 int glite::Crypt::m_bufferSize [private]

buffer size to encrypt/decrypt file

6.3.5.2 const std::string glite::Crypt::m_cipherName [private]

name of OpenSSL cypher type

6.3.5.3 EVP_CIPHER_CTX* glite::Crypt::m_dctx [private]

OpenSSL decrypting cipher context.

6.3.5.4 EVP_CIPHER_CTX* glite::Crypt::m_ectx [private]

OpenSSL encrypting cipher context.

6.3.5.5 Key glite::Crypt::m_iv [private]

initialisation vector

6.3.5.6 unsigned int glite::Crypt::m_ivLength [private]

initialisation vector length

6.3.5.7 Key glite::Crypt::m_key [private]

encryption/decryption key

6.3.5.8 unsigned int glite::Crypt::m_keyLength [private]

encryption/decryption key length

6.3.5.9 const EVP_CIPHER* glite::Crypt::m_type [private]

OpenSSL EVP_CIPHER structure.

6.3.5.10 int glite::Crypt::m_verbose [private]

verbose level

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

- src/crypt.h
- src/base64.cc
- src/crypt.cc
- src/io.cc
- src/shamir.cc
- src/tss.cc

6.4 glite::inputKey< key_type > Struct Template Reference

`#include <crypt.h>`

6.4.1 Detailed Description

template<typename key_type> struct glite::inputKey< key_type >

State structure manipulator to input key.

Public Member Functions

- inputKey (const char separator=0)
- ~inputKey ()

Public Attributes

- key_type key_
- char separator_

6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<typename key_type> glite::inputKey< key_type >::inputKey (const char *separator* = 0)

6.4.2.2 template<typename key_type> glite::inputKey< key_type >::~inputKey ()

6.4.3 Member Data Documentation

6.4.3.1 template<typename key_type> key_type glite::inputKey< key_type >::key_

6.4.3.2 template<typename key_type> char glite::inputKey< key_type >::separator_

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

- src/crypt.h
- src/io.cc

6.5 glite::outputKey< key_type > Struct Template Reference

`#include <crypt.h>`

6.5.1 Detailed Description

template<typename key_type> struct glite::outputKey< key_type >

State structure manipulator to output key.

Public Member Functions

- outputKey (const key_type &key, const char separator=0, int lineWidth=64)
- ~outputKey ()

Public Attributes

- key_type key_
- char separator_
- int lineWidth_

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<typename key_type> glite::outputKey< key_type >::outputKey (const key_type & *key*, const char *separator* = 0, int *lineWidth* = 64)

6.5.2.2 template<typename key_type> glite::outputKey< key_type >::~outputKey ()

6.5.3 Member Data Documentation

6.5.3.1 template<typename key_type> key_type glite::outputKey< key_type >::key_

6.5.3.2 template<typename key_type> int glite::outputKey< key_type >::lineWidth_

6.5.3.3 template<typename key_type> char glite::outputKey< key_type >::separator_

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

- src/crypt.h
- src/io.cc

6.6 glite::setFormat Struct Reference

`#include <crypt.h>`

6.6.1 Detailed Description

State structure manipulator to format key.

Public Member Functions

- setFormat (std::ios::fmtflags base, int width=0, char padding=' ')
- ~setFormat ()

Public Attributes

- std::ios::fmtflags base_
- int width_
- char padding_

6.6.2 Constructor & Destructor Documentation

6.6.2.1 glite::setFormat::setFormat (std::ios::fmtflags *base*, int *width* = 0, char *padding* = ' ')

6.6.2.2 glite::setFormat::~setFormat ()

6.6.3 Member Data Documentation

6.6.3.1 std::ios::fmtflags glite::setFormat::base_

6.6.3.2 char glite::setFormat::padding_

6.6.3.3 int glite::setFormat::width_

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

- src/crypt.h
- src/io.cc

7 Glite Security encrypted storage cpp File Documentation

7.1 src/base64.cc File Reference

7.1.1 Detailed Description

Definitions of member functions for Base64 encoding/decoding.

Uses OpenSSL BIO interface.

test-base64.cc This is a simple test for Base64 encoding/decoding

```
#include <crypt.h>
```

Namespaces

- namespace glite

7.2 src/crypt.cc File Reference

7.2.1 Detailed Description

Definitions of member functions for Pseudo-random key and initialisation vector generation Encryption and decryption blocks and files Miscelleanous modifiers and queries Error handling.

- ∗ - ∗ - ∗ - Standalone functions to convert from/to key to/from array block

Uses OpenSSL EVP cipher and ERR interfaces.

```
#include "crypt.h"
```

Namespaces

- namespace glite

7.3 src/crypt.h File Reference

7.3.1 Detailed Description

Declarations for gLite encryted data storage.

Provides declarations for a set of functionalities to process files

- buffer and file encryption/decryption,

- key generation
- key manipulation
  - split/join
  - base64-encode/decode
- key i/o manipulation.

This file is the only file to include.

```
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <stdexcept>
#include <vector>
#include <limits>
#include <openssl/bio.h>
#include <openssl/evp.h>
#include <openssl/err.h>
#include <openssl/rand.h>
```

Namespaces

- namespace glite

Defines

- #define CRYPT_H 1

7.3.2 Define Documentation

7.3.2.1 #define CRYPT_H 1

7.4 src/io.cc File Reference

7.4.1 Detailed Description

Definitions for key I/O manipulation.

test-base64.cc test-tss.cc test-shamir.cc

```
#include <crypt.h>
```

Namespaces

- namespace glite

7.5 src/shamir.cc File Reference

7.5.1 Detailed Description

Definition for Shamir secret sharing (SSS) scheme.

Based on "How to Share a Secret", by Adi Shamir, Communications of the ACM, November, 1979, Volume 22, Number 11, page 612.

SSS provides a perfect (t, n)-threshold secret sharing scheme. i.e. it is a method for n parties to carry shares s_i of a secret s such that any t of them are needed to recover the secret, but so that no $t - 1$ of them can do so. The threshold is perfect if knowledge of $t - 1$ or fewer shares provides no information regarding s. Shamir (t, n)-threshold scheme is based on classical Lagrange polynomial interpolation of degree $t - 1$ with modular arithmetic instead of real arithmetic. The set of integers modulo a prime number 'prime' forms a field in which interpolation is possible. We choose to break the key in byte unit handled separately. The prime number has to be bigger than the largest number than can be represented (i.e. 0xff) and the number of shares n. In order to handle all byte values (0x00-0xff) and so that we can multiply two such numbers, the shared secrets will fit in a 16-bits integer (unsigned short). The prime has to be less than 2^16 in order to fit them all. The largest such prime is 65521.

test-shamir.cc This is an simple split/join test for Shamir secret sharing scheme.

```
#include <crypt.h>
```

Namespaces

- namespace glite

7.6 src/test-base64.cc File Reference

```
#include <crypt.h>
```

Functions

- void usage (const char pname[ ])
- int main (int nargs, const char *args[ ])

7.6.1 Function Documentation

7.6.1.1 int main (int *nargs*, const char * *args*[ ])

7.6.1.2 void usage (const char *pname*[ ])

7.7 src/test-shamir.cc File Reference

```
#include <crypt.h>
```

Functions

- void usage (const char pname[ ])
- int main (int nargs, const char *args[ ])

7.7.1 Function Documentation

7.7.1.1 int main (int *nargs*, const char * *args*[ ])

7.7.1.2 void usage (const char *pname*[ ])

7.8 src/test-tss.cc File Reference

```
#include <crypt.h>
```

Functions

- void usage (const char pname[ ])
- int main (int nargs, const char *args[ ])

7.8.1 Function Documentation

7.8.1.1 int main (int *nargs*, const char * *args*[ ])

7.8.1.2 void usage (const char *pname*[ ])

7.9 src/tss.cc File Reference

7.9.1 Detailed Description

Definitions for Trivial secret sharing (TSS) scheme.

Based on the Wikipedia http://en.wikipedia.org/wiki/Secret_sharing

test-tss.cc This is a simple test for Trivial secret sharing

```
#include <crypt.h>
```

Namespaces

- namespace glite

Index

~Crypt
    glite::Crypt, 13
~inputKey
    glite::inputKey, 20
~outputKey
    glite::outputKey, 21
~setFormat
    glite::setFormat, 22

Array2Key
    glite, 4

Base64Key
    glite::Crypt, 12
base_
    glite::setFormat, 22
byte
    glite::Crypt, 12

Crypt
    glite::Crypt, 13
crypt.h
    CRYPT_H, 24
CRYPT_H
    crypt.h, 24

decodeBase64
    glite::Crypt, 13
decrypt
    glite::Crypt, 13
divideModulo
    glite::Crypt, 14
drawIV
    glite::Crypt, 14
drawKey
    glite::Crypt, 14
drawPRN
    glite::Crypt, 14

encodeBase64
    glite::Crypt, 14
encrypt
    glite::Crypt, 14, 15
evalLagrangeInterpAt0
    glite::Crypt, 15
evalPolynom
    glite::Crypt, 15
evp_max_block_length
    glite, 7

getBufferSize
    glite::Crypt, 15
getIV
    glite::Crypt, 15
getKey
    glite::Crypt, 15
getVerbose
    glite::Crypt, 15
glite, 2
    Array2Key, 4
    evp_max_block_length, 7
    Key2Array, 4, 5
    operator<<, 5, 6
    operator>>, 6
    prime, 7
    transform, 6
glite::assignXor, 7
glite::assignXor
    operator(), 7
glite::bitwiseXor, 7
glite::bitwiseXor
    operator(), 8
glite::Crypt, 8
    ~Crypt, 13
    Base64Key, 12
    byte, 12
    Crypt, 13
    decodeBase64, 13
    decrypt, 13
    divideModulo, 14
    drawIV, 14
    drawKey, 14
    drawPRN, 14
    encodeBase64, 14
    encrypt, 14, 15
    evalLagrangeInterpAt0, 15
    evalPolynom, 15
    getBufferSize, 15
    getIV, 15
    getKey, 15
    getVerbose, 15
    handleError, 16
    initCrypters, 16
    initPolynom, 16
    inverseModulo, 16
    joinKeySSS, 16
    joinKeyTSS, 16
    Key, 12
    longByte, 12
    LongKey, 12
    m_bufferSize, 18
    m_cipherName, 18

m_dctx, 18
m_ectx, 18
m_iv, 18
m_ivLength, 18
m_key, 18
m_keyLength, 18
m_type, 19
m_verbose, 19
Polynom, 12
printOn, 17
seedPRNG, 17
setBufferSize, 17
setKeyAndIV, 17
setVerbose, 17
shortByte, 12
ShortKey, 12
SplitKeys, 12
splitKeySSS, 17
splitKeyTSS, 17
SplitShortKeys, 12
glite::inputKey, 19
glite::inputKey
~inputKey, 20
inputKey, 19
key_, 20
separator_, 20
glite::outputKey, 20
glite::outputKey
~outputKey, 21
key_, 21
lineWidth_, 21
outputKey, 20
separator_, 21
glite::setFormat, 21
glite::setFormat
~setFormat, 22
base_, 22
padding_, 22
setFormat, 22
width_, 22

handleError
glite::Crypt, 16

initCrypters
glite::Crypt, 16
initPolynom
glite::Crypt, 16
inputKey
glite::inputKey, 19
inverseModulo
glite::Crypt, 16

joinKeySSS
glite::Crypt, 16
joinKeyTSS
glite::Crypt, 16

Key
glite::Crypt, 12
Key2Array
glite, 4, 5
key_
glite::inputKey, 20
glite::outputKey, 21

lineWidth_
glite::outputKey, 21
longByte
glite::Crypt, 12
LongKey
glite::Crypt, 12

m_bufferSize
glite::Crypt, 18
m_cipherName
glite::Crypt, 18
m_dctx
glite::Crypt, 18
m_ectx
glite::Crypt, 18
m_iv
glite::Crypt, 18
m_ivLength
glite::Crypt, 18
m_key
glite::Crypt, 18
m_keyLength
glite::Crypt, 18
m_type
glite::Crypt, 19
m_verbose
glite::Crypt, 19
main
test-base64.cc, 25
test-shamir.cc, 25
test-tss.cc, 26

operator()
glite::assignXor, 7
glite::bitwiseXor, 8
operator<<
glite, 5, 6
operator>>
glite, 6
outputKey
glite::outputKey, 20

padding_

glite::setFormat, 22
Polynom
glite::Crypt, 12
prime
glite, 7
printOn
glite::Crypt, 17

seedPRNG
glite::Crypt, 17
separator_
glite::inputKey, 20
glite::outputKey, 21
setBufferSize
glite::Crypt, 17
setFormat
glite::setFormat, 22
setKeyAndIV
glite::Crypt, 17
setVerbose
glite::Crypt, 17
shortByte
glite::Crypt, 12
ShortKey
glite::Crypt, 12
SplitKeys
glite::Crypt, 12
splitKeySSS
glite::Crypt, 17
splitKeyTSS
glite::Crypt, 17
SplitShortKeys
glite::Crypt, 12
src/base64.cc, 22
src/crypt.cc, 23
src/crypt.h, 23
src/io.cc, 24
src/shamir.cc, 24
src/test-base64.cc, 25
src/test-shamir.cc, 25
src/test-tss.cc, 26
src/tss.cc, 26

test-base64.cc
main, 25
usage, 25
test-shamir.cc
main, 25
usage, 26
test-tss.cc
main, 26
usage, 26
transform
glite, 6

usage
test-base64.cc, 25
test-shamir.cc, 26
test-tss.cc, 26

width_
glite::setFormat, 22