SA104X

Generated by Doxygen 1.8.3.1

Mon Mar 25 2013 11:20:34

Contents

Chapter 1

sa104x-kexjobb

Implementation of a verifier for the Verificatum Mix-net

Requirements

- OpenSSL
- GMP
- RapidXML

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

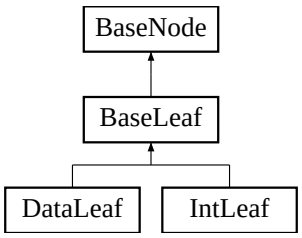
Chapter 5

Class Documentation

5.1 BaseLeaf Class Reference

`#include <BaseLeaf.h>`

Inheritance diagram for BaseLeaf:



Public Member Functions

- BaseLeaf (BaseLeaf::NodeType type)
- ~BaseLeaf (void)

Additional Inherited Members

5.1.1 Constructor & Destructor Documentation

5.1.1.1 BaseLeaf::BaseLeaf (BaseLeaf::NodeType *type*) [explicit]

5.1.1.2 BaseLeaf::~BaseLeaf (void)

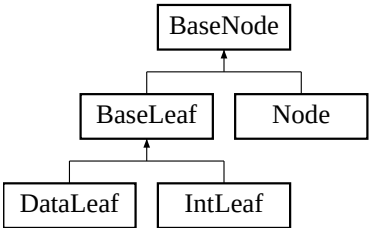
The documentation for this class was generated from the following files:

- Arithmetic/BaseLeaf.h
- Arithmetic/BaseLeaf.cpp

5.2 BaseNode Class Reference

`#include <BaseNode.h>`

Inheritance diagram for BaseNode:



Public Types

- enum NodeType { NODE = 0, INT_LEAF = 1, DATA_LEAF = 2 }

Public Member Functions

- BaseNode (BaseNode::NodeType type)
- ~BaseNode (void)
- BaseNode::NodeType getType (void) const
- virtual int32_t getLength (void) const =0
- virtual bytevector toVector (void) const =0
- bytevector serialize () const
- bytevector concatData (const BaseNode *const other) const

Static Protected Member Functions

- static BaseNode * copy (const BaseNode *node)
- static void ReadNodeHeader (std::istream &file, char &type, uint32_t &length)

5.2.1 Detailed Description

The basic node from which all nodes in a Byte Tree inherit.

5.2.2 Member Enumeration Documentation

5.2.2.1 enum BaseNode::NodeType

Enumerator

NODE A node which contains other nodes.

INT_LEAF An IntLeaf which contains a number.

DATA_LEAF A DataLeaf which contains a string or vector of bytes.

5.2.3 Constructor & Destructor Documentation

5.2.3.1 BaseNode::BaseNode (BaseNode::NodeType *type*)

5.2.3.2 BaseNode::~BaseNode (void)

5.2.4 Member Function Documentation

5.2.4.1 bytevector BaseNode::concatData (const BaseNode *const *other*) const

5.2.4.2 BaseNode * BaseNode::copy (const BaseNode * *node*) [static], [protected]

5.2.4.3 virtual int32_t BaseNode::getLength (void) const [pure virtual]

Implemented in Node, IntLeaf, and DataLeaf.

5.2.4.4 BaseNode::NodeType BaseNode::getType (void) const

5.2.4.5 void BaseNode::ReadNodeHeader (std::istream & *file*, char & *type*, uint32_t & *length*) [static], [protected]

5.2.4.6 bytevector BaseNode::serialize ( ) const

5.2.4.7 virtual bytevector BaseNode::toVector (void) const [pure virtual]

Implemented in IntLeaf, Node, and DataLeaf.

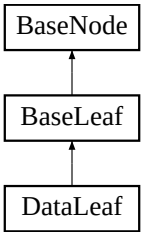
The documentation for this class was generated from the following files:

- Arithmetic/BaseNode.h
- Arithmetic/BaseNode.cpp

5.3 DataLeaf Class Reference

`#include <DataLeaf.h>`

Inheritance diagram for DataLeaf:



Public Member Functions

- DataLeaf (void)
- DataLeaf (int32_t size)
- DataLeaf (std::istream &file)
- DataLeaf (std::string str)
- ~DataLeaf (void)
- bytevector & getData (void)
- const bytevector & getData (void) const
- virtual bytevector toVector (void) const
- virtual int32_t getLength (void) const
- DataLeaf & operator= (const DataLeaf &leaf)

Additional Inherited Members

5.3.1 Constructor & Destructor Documentation

5.3.1.1 DataLeaf::DataLeaf (void)

5.3.1.2 DataLeaf::DataLeaf (int32_t *size*) [explicit]

5.3.1.3 DataLeaf::DataLeaf (std::istream & *file*)

5.3.1.4 DataLeaf::DataLeaf (std::string *str*)

5.3.1.5 DataLeaf::~DataLeaf (void)

5.3.2 Member Function Documentation

5.3.2.1 bytevector & DataLeaf::getData (void)

5.3.2.2 const bytevector & DataLeaf::getData (void) const

5.3.2.3 int32_t DataLeaf::getLength (void) const [virtual]

Implements BaseNode.

5.3.2.4 DataLeaf & DataLeaf::operator= (const DataLeaf & *leaf*)

5.3.2.5 bytevector DataLeaf::toVector (void) const [virtual]

Implements BaseNode.

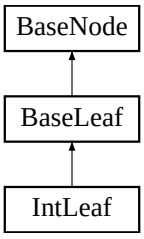
The documentation for this class was generated from the following files:

- Arithmetic/DataLeaf.h
- Arithmetic/DataLeaf.cpp

5.4 IntLeaf Class Reference

`#include <IntLeaf.h>`

Inheritance diagram for IntLeaf:



Public Member Functions

- IntLeaf (void)
- IntLeaf (const IntLeaf &leaf)
- IntLeaf (const mpz_class &bigint)
- IntLeaf (long int input)
- IntLeaf (long int input, long int length)
- IntLeaf (std::string input)
- IntLeaf (bytevector bytevec)

- IntLeaf (std::istream &file)
- ~IntLeaf (void)
- IntLeaf & operator= (const IntLeaf &leaf)
- IntLeaf & operator= (long int input)
- IntLeaf & operator= (std::string input)
- IntLeaf & modTo (const IntLeaf &leaf)
- IntLeaf mod (const IntLeaf &leaf) const
- IntLeaf & addTo (const IntLeaf &leaf)
- IntLeaf add (const IntLeaf &leaf) const
- IntLeaf & addToMod (const IntLeaf &leaf, const IntLeaf &mod)
- IntLeaf addMod (const IntLeaf &leaf, const IntLeaf &mod) const
- IntLeaf & operator+= (const IntLeaf &leaf)
- IntLeaf operator+ (const IntLeaf &leaf) const
- IntLeaf & multTo (const IntLeaf &leaf)
- IntLeaf mult (const IntLeaf &leaf) const
- IntLeaf & multToMod (const IntLeaf &leaf, const IntLeaf &mod)
- IntLeaf multMod (const IntLeaf &leaf, const IntLeaf &mod) const
- IntLeaf & operator*= (const IntLeaf &leaf)
- IntLeaf operator* (const IntLeaf &leaf) const
- IntLeaf & expTo (unsigned long exponent)
- IntLeaf exp (unsigned long exponent) const
- IntLeaf & expToMod (const IntLeaf &leaf, const IntLeaf &mod)
- IntLeaf expMod (const IntLeaf &leaf, const IntLeaf &mod) const
- bool operator== (const IntLeaf &leaf) const
- bool operator!= (const IntLeaf &leaf) const
- bool operator< (const IntLeaf &leaf) const
- bool operator> (const IntLeaf &leaf) const
- IntLeaf operator- (void) const
- IntLeaf inverse (const IntLeaf &mod) const
- mpz_class getBigInt (void) const
- virtual bytevector toVector (void) const
- virtual int32_t getLength (void) const
- std::string toString (void) const

Static Public Member Functions

- static BaseNode * constructPartFromFile (std::istream &file, uint32_t length)

Static Public Attributes

- static const int ARRAYORDER = 1
- static const int ENDIAN = 0
- static const int NAILS = 0

Additional Inherited Members

5.4.1 Constructor & Destructor Documentation

5.4.1.1 IntLeaf::IntLeaf (void)

5.4.1.2 IntLeaf::IntLeaf (const IntLeaf & *leaf*)

5.4.1.3 IntLeaf::IntLeaf (const mpz_class & *bigint*) [explicit]

5.4.1.4 IntLeaf::IntLeaf (long int *input*)

5.4.1.5 IntLeaf::IntLeaf (long int *input*, long int *length*)

5.4.1.6 IntLeaf::IntLeaf (std::string *input*) [explicit]

5.4.1.7 IntLeaf::IntLeaf (**bytevector** *bytevec*) [explicit]

5.4.1.8 IntLeaf::IntLeaf (std::istream & *file*) [explicit]

5.4.1.9 IntLeaf::~IntLeaf (void)

5.4.2 Member Function Documentation

5.4.2.1 **IntLeaf** IntLeaf::add (const **IntLeaf** & *leaf*) const

5.4.2.2 **IntLeaf** IntLeaf::addMod (const **IntLeaf** & *leaf*, const **IntLeaf** & *mod*) const

5.4.2.3 **IntLeaf** & IntLeaf::addTo (const **IntLeaf** & *leaf*)

5.4.2.4 **IntLeaf** & IntLeaf::addToMod (const **IntLeaf** & *leaf*, const **IntLeaf** & *mod*)

5.4.2.5 **BaseNode** * IntLeaf::constructPartFromFile (std::istream & *file*, uint32_t *length*) [static]

5.4.2.6 **IntLeaf** IntLeaf::exp (unsigned long *exponent*) const

5.4.2.7 **IntLeaf** IntLeaf::expMod (const **IntLeaf** & *leaf*, const **IntLeaf** & *mod*) const

5.4.2.8 **IntLeaf** & IntLeaf::expTo (unsigned long *exponent*)

5.4.2.9 **IntLeaf** & IntLeaf::expToMod (const **IntLeaf** & *leaf*, const **IntLeaf** & *mod*)

5.4.2.10 mpz_class IntLeaf::getBigInt (void) const

5.4.2.11 int32_t IntLeaf::getLength (void) const [virtual]

Implements BaseNode.

5.4.2.12 **IntLeaf** IntLeaf::inverse (const **IntLeaf** & *mod*) const

5.4.2.13 **IntLeaf** IntLeaf::mod (const **IntLeaf** & *leaf*) const

5.4.2.14 **IntLeaf** & IntLeaf::modTo (const **IntLeaf** & *leaf*)

5.4.2.15 **IntLeaf** IntLeaf::mult (const **IntLeaf** & *leaf*) const

5.4.2.16 **IntLeaf** IntLeaf::multMod (const **IntLeaf** & *leaf*, const **IntLeaf** & *mod*) const

5.4.2.17 **IntLeaf** & IntLeaf::multTo (const **IntLeaf** & *leaf*)

5.4.2.18 **IntLeaf** & IntLeaf::multToMod (const **IntLeaf** & *leaf*, const **IntLeaf** & *mod*)

5.4.2.19 bool IntLeaf::operator!= (const **IntLeaf** & *leaf*) const

5.4.2.20 **IntLeaf** IntLeaf::operator∗ (const **IntLeaf** & *leaf*) const

5.4.2.21 **IntLeaf** & IntLeaf::operator∗= (const **IntLeaf** & *leaf*)

5.4.2.22 **IntLeaf** IntLeaf::operator+ (const **IntLeaf** & *leaf*) const

5.4.2.23 **IntLeaf** & IntLeaf::operator+= (const **IntLeaf** & *leaf*)

5.4.2.24 **IntLeaf** IntLeaf::operator- (void) const

5.4.2.25 bool IntLeaf::operator< (const **IntLeaf** & *leaf*) const

5.4.2.26 **IntLeaf** & IntLeaf::operator= (const **IntLeaf** & *leaf*)

5.4.2.27 **IntLeaf** & IntLeaf::operator= (long int *input*)

5.4.2.28 **IntLeaf** & IntLeaf::operator= (std::string *input*)

5.4.2.29 bool IntLeaf::operator== (const **IntLeaf** & *leaf*) const

5.4.2.30 bool IntLeaf::operator> (const **IntLeaf** & *leaf*) const

5.4.2.31 std::string IntLeaf::toString (void) const

5.4.2.32 **bytevector** IntLeaf::toVector (void) const [virtual]

Implements BaseNode.

5.4.3 Member Data Documentation

5.4.3.1 const int IntLeaf::ARRAYORDER = 1 [static]

5.4.3.2 const int IntLeaf::ENDIAN = 0 [static]

5.4.3.3 const int IntLeaf::NAILS = 0 [static]

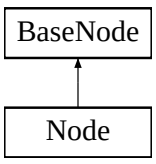
The documentation for this class was generated from the following files:

- Arithmetic/IntLeaf.h
- Arithmetic/IntLeaf.cpp

5.5 Node Class Reference

`#include <Node.h>`

Inheritance diagram for Node:



Public Member Functions

- Node (void)
- Node (const Node &node)
- Node (const bytevector data)
- Node (const std::string filename)
- Node (std::istream &file)
- ~Node (void)
- virtual bytevector toVector (void) const
- Node & operator= (const Node &node)
- Node & modTo (const IntLeaf &leaf)
- Node mod (const IntLeaf &leaf) const
- Node & addTo (const IntLeaf &leaf)
- Node add (const IntLeaf &leaf) const
- Node & addTo (const Node &node)
- Node add (const Node &node) const
- Node & addToMod (const IntLeaf &leaf, const IntLeaf &mod)
- Node addMod (const IntLeaf &leaf, const IntLeaf &mod) const
- Node & addToMod (const Node &node, const IntLeaf &mod)
- Node addMod (const Node &node, const IntLeaf &mod) const
- Node & operator+= (const IntLeaf &leaf)
- Node operator+ (const IntLeaf &leaf) const
- Node & operator+= (const Node &node)
- Node operator+ (const Node &node) const
- Node & multTo (const IntLeaf &leaf)
- Node mult (const IntLeaf &leaf) const
- Node & multTo (const Node &node)
- Node mult (const Node &node) const
- Node & multToMod (const IntLeaf &leaf, const IntLeaf &mod)
- Node multMod (const IntLeaf &leaf, const IntLeaf &mod) const
- Node & multToMod (const Node &node, const IntLeaf &mod)
- Node multMod (const Node &node, const IntLeaf &mod) const
- Node & operator*= (const IntLeaf &leaf)
- Node operator* (const IntLeaf &leaf) const
- Node & operator*= (const Node &node)
- Node operator* (const Node &node) const
- bool operator== (const Node &node) const
- bool operator!= (const Node &node) const
- IntLeaf sum (void) const
- IntLeaf sumMod (const IntLeaf &mod) const
- IntLeaf prod (void) const
- IntLeaf prodMod (const IntLeaf &mod) const
- Node exp (unsigned long exponent) const
- Node expMod (unsigned long exponent, const IntLeaf &mod) const
- Node expMod (const IntLeaf &exponent, const IntLeaf &mod) const
- Node exp (const Node &exponents) const
- Node expMod (const Node &exponents, const IntLeaf &mod) const
- Node & expTo (unsigned long exponent)
- Node & expToMod (unsigned long exponent, const IntLeaf &mod)
- Node & expToMod (const IntLeaf &exponent, const IntLeaf &mod)
- Node & expTo (const Node &exponents)
- Node & expToMod (const Node &exponents, const IntLeaf &mod)
- IntLeaf expMultMod (const Node &node, const IntLeaf &mod) const
- IntLeaf expMult (unsigned long exponent) const
- IntLeaf expMultMod (unsigned long exponent, const IntLeaf &mod) const

- Node & addChild (const BaseNode &child)
- Node getChildren (int32_t index) const
- IntLeaf & getIntLeafChild (int32_t index)
- const IntLeaf & getIntLeafChild (int32_t index) const
- Node & getNodeChild (int32_t index)
- const Node & getNodeChild (int32_t index) const
- std::string toString (void) const
- virtual int32_t getLength (void) const

Static Public Member Functions

- static BaseNode * constructPartFromFile (std::istream &file, uint32_t count)

Additional Inherited Members

5.5.1 Constructor & Destructor Documentation

5.5.1.1 Node::Node (void)

5.5.1.2 Node::Node (const Node & *node*)

5.5.1.3 Node::Node (const bytevector *data*)

5.5.1.4 Node::Node (const std::string *filename*) [explicit]

5.5.1.5 Node::Node (std::istream & *file*) [explicit]

5.5.1.6 Node::~Node (void)

5.5.2 Member Function Documentation

5.5.2.1 Node Node::add (const IntLeaf & *leaf*) const

5.5.2.2 Node Node::add (const Node & *node*) const

5.5.2.3 Node & Node::addChild (const BaseNode & *child*)

5.5.2.4 Node Node::addMod (const IntLeaf & *leaf,* const IntLeaf & *mod*) const

5.5.2.5 Node Node::addMod (const Node & *node,* const IntLeaf & *mod*) const

5.5.2.6 Node & Node::addTo (const IntLeaf & *leaf*)

5.5.2.7 Node & Node::addTo (const Node & *node*)

5.5.2.8 Node & Node::addToMod (const IntLeaf & *leaf,* const IntLeaf & *mod*)

5.5.2.9 Node & Node::addToMod (const Node & *node,* const IntLeaf & *mod*)

5.5.2.10 BaseNode * Node::constructPartFromFile (std::istream & *file,* uint32_t *count*) [static]

5.5.2.11 Node Node::exp (unsigned long *exponent*) const

5.5.2.12 Node Node::exp (const Node & *exponents*) const

5.5.2.13 **Node** Node::expMod (unsigned long *exponent,* const **IntLeaf** & *mod*) const

5.5.2.14 **Node** Node::expMod (const **IntLeaf** & *exponent,* const **IntLeaf** & *mod*) const

5.5.2.15 **Node** Node::expMod (const **Node** & *exponents,* const **IntLeaf** & *mod*) const

5.5.2.16 **IntLeaf** Node::expMult (unsigned long *exponent*) const

5.5.2.17 **IntLeaf** Node::expMultMod (const **Node** & *node,* const **IntLeaf** & *mod*) const

5.5.2.18 **IntLeaf** Node::expMultMod (unsigned long *exponent,* const **IntLeaf** & *mod*) const

5.5.2.19 **Node** & Node::expTo (unsigned long *exponent*)

5.5.2.20 **Node**& Node::expTo (const **Node** & *exponents*)

5.5.2.21 **Node** & Node::expToMod (unsigned long *exponent,* const **IntLeaf** & *mod*)

5.5.2.22 **Node** & Node::expToMod (const **IntLeaf** & *exponent,* const **IntLeaf** & *mod*)

5.5.2.23 **Node**& Node::expToMod (const **Node** & *exponents,* const **IntLeaf** & *mod*)

5.5.2.24 **Node** Node::getChildren (int32_t *index*) const

5.5.2.25 **IntLeaf** & Node::getIntLeafChild (int32_t *index*)

5.5.2.26 const **IntLeaf** & Node::getIntLeafChild (int32_t *index*) const

5.5.2.27 int32_t Node::getLength (void) const [virtual]

Implements BaseNode.

5.5.2.28 **Node** & Node::getNodeChild (int32_t *index*)

5.5.2.29 const **Node** & Node::getNodeChild (int32_t *index*) const

5.5.2.30 **Node** Node::mod (const **IntLeaf** & *leaf*) const

5.5.2.31 **Node** & Node::modTo (const **IntLeaf** & *leaf*)

5.5.2.32 **Node** Node::mult (const **IntLeaf** & *leaf*) const

5.5.2.33 **Node** Node::mult (const **Node** & *node*) const

5.5.2.34 **Node** Node::multMod (const **IntLeaf** & *leaf,* const **IntLeaf** & *mod*) const

5.5.2.35 **Node** Node::multMod (const **Node** & *node,* const **IntLeaf** & *mod*) const

5.5.2.36 **Node** & Node::multTo (const **IntLeaf** & *leaf*)

5.5.2.37 **Node** & Node::multTo (const **Node** & *node*)

5.5.2.38 **Node** & Node::multToMod (const **IntLeaf** & *leaf,* const **IntLeaf** & *mod*)

5.5.2.39 **Node** & Node::multToMod (const **Node** & *node,* const **IntLeaf** & *mod*)

5.5.2.40 bool Node::operator!= (const Node & *node*) const

5.5.2.41 Node Node::operator* (const IntLeaf & *leaf*) const

5.5.2.42 Node Node::operator* (const Node & *node*) const

5.5.2.43 Node & Node::operator*= (const IntLeaf & *leaf*)

5.5.2.44 Node & Node::operator*= (const Node & *node*)

5.5.2.45 Node Node::operator+ (const IntLeaf & *leaf*) const

5.5.2.46 Node Node::operator+ (const Node & *node*) const

5.5.2.47 Node & Node::operator+= (const IntLeaf & *leaf*)

5.5.2.48 Node& Node::operator+= (const Node & *node*)

5.5.2.49 Node & Node::operator= (const Node & *node*)

5.5.2.50 bool Node::operator== (const Node & *node*) const

5.5.2.51 IntLeaf Node::prod (void) const

5.5.2.52 IntLeaf Node::prodMod (const IntLeaf & *mod*) const

5.5.2.53 IntLeaf Node::sum (void) const

5.5.2.54 IntLeaf Node::sumMod (const IntLeaf & *mod*) const

5.5.2.55 std::string Node::toString (void) const

5.5.2.56 bytevector Node::toVector (void) const [virtual]

Implements BaseNode.

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

- Arithmetic/Node.h
- Arithmetic/Node.cpp

5.6 PRG Class Reference

`#include <PRG.h>`

Public Member Functions

- PRG (bytevector(*hash)(bytevector data), bytevector seed, unsigned int outbits)
- ~PRG (void)
- IntLeaf next ()

5.6.1 Constructor & Destructor Documentation

5.6.1.1 PRG::PRG (bytevector(∗)(bytevector data) *hash,* bytevector *seed,* unsigned int *outbits*)

5.6.1.2 PRG::∼PRG (void)

5.6.2 Member Function Documentation

5.6.2.1 IntLeaf PRG::next ( )

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

- Crypto/PRG.h
- Crypto/PRG.cpp

5.7 proofStruct Struct Reference

`#include <Utilities.h>`

Public Attributes

- IntLeaf rho
- unsigned int N
- unsigned int lambda
- unsigned int width
- unsigned int nE
- unsigned int nR
- unsigned int nV
- unsigned int nHash
- bytevector(∗ hash)(bytevector)
- Node Gq
- Node Rw
- Node pk
- Node y
- Node x

5.7.1 Member Data Documentation

5.7.1.1 Node proofStruct::Gq

5.7.1.2 bytevector(∗ proofStruct::hash)(bytevector)

5.7.1.3 unsigned int proofStruct::lambda

5.7.1.4 unsigned int proofStruct::N

5.7.1.5 unsigned int proofStruct::nE

5.7.1.6 unsigned int proofStruct::nHash

5.7.1.7 unsigned int proofStruct::nR

5.7.1.8 unsigned int proofStruct::nV

5.7.1.9 Node proofStruct::pk

5.7.1.10 IntLeaf proofStruct::rho

5.7.1.11 Node proofStruct::Rw

5.7.1.12 unsigned int proofStruct::width

5.7.1.13 Node proofStruct::x

5.7.1.14 Node proofStruct::y

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

- Verifier/Utilities.h

5.8 RO Class Reference

`#include <RO.h>`

Public Member Functions

- RO (bytevector(∗hash)(bytevector data), int Nout)
- ∼RO (void)
- IntLeaf operator() (bytevector data)

5.8.1 Constructor & Destructor Documentation

5.8.1.1 RO::RO (bytevector(∗)(bytevector data) *hash,* int *Nout*)

5.8.1.2 RO::∼RO (void)

5.8.2 Member Function Documentation

5.8.2.1 IntLeaf RO::operator() (bytevector *data*)

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

- Crypto/RO.h
- Crypto/RO.cpp

Chapter 6

File Documentation

6.1 Arithmetic/BaseLeaf.cpp File Reference

```
#include "BaseLeaf.h"
```

6.2 Arithmetic/BaseLeaf.h File Reference

```
#include "basenode.h"
#include <vector>
```

Classes

- class BaseLeaf

6.3 Arithmetic/BaseNode.cpp File Reference

```
#include "BaseNode.h"
#include "Node.h"
#include "IntLeaf.h"
#include "DataLeaf.h"
```

6.4 Arithmetic/BaseNode.h File Reference

```
#include <stdint.h>
#include <istream>
#include "types.h"
```

Classes

- class BaseNode

6.5 Arithmetic/DataLeaf.cpp File Reference

```
#include "DataLeaf.h"
#include <stdexcept>
```

Macros

- #define ARRAYORDER -1 /∗ -1 for least significant first, 1 for most significant first ∗/

6.5.1 Macro Definition Documentation

6.5.1.1 #define ARRAYORDER -1 /∗ -1 for least significant first, 1 for most significant first ∗/

6.6 Arithmetic/DataLeaf.h File Reference

```
#include "baseleaf.h"
#include <vector>
#include <istream>
#include <string>
```

Classes

- class DataLeaf

6.7 Arithmetic/IntLeaf.cpp File Reference

```
#include "IntLeaf.h"
```

6.8 Arithmetic/IntLeaf.h File Reference

```
#include "BaseLeaf.h"
#include <gmp.h>
#include <gmpxx.h>
#include <string>
#include <fstream>
```

Classes

- class IntLeaf

6.9 Arithmetic/Node.cpp File Reference

```
#include "Node.h"
#include "DataLeaf.h"
#include <sstream>
#include <iomanip>
#include <stdlib.h>
#include <iterator>
#include <iostream>
```

6.10 Arithmetic/Node.h File Reference

```
#include "BaseNode.h"
#include "IntLeaf.h"
#include <vector>
#include <fstream>
#include <string>
```

Classes

- class Node

6.11 Arithmetic/types.h File Reference

```
#include <vector>
```

Typedefs

- typedef std::vector< unsigned char > bytevector

6.11.1 Typedef Documentation

6.11.1.1 typedef std::vector<unsigned char> bytevector

6.12 Crypto/ElGamal.cpp File Reference

```
#include "ElGamal.h"
```

Functions

- IntLeaf PDec (IntLeaf x, IntLeaf u, IntLeaf mod)
- IntLeaf TDec (IntLeaf f, IntLeaf v, IntLeaf mod)
- Node Enc (Node pk, IntLeaf m, IntLeaf s, IntLeaf mod)

6.12.1 Function Documentation

6.12.1.1 Node Enc (Node *pk,* IntLeaf *m,* IntLeaf *s,* IntLeaf *mod*)

6.12.1.2 IntLeaf PDec (IntLeaf *x,* IntLeaf *u,* IntLeaf *mod*)

6.12.1.3 IntLeaf TDec (IntLeaf *f,* IntLeaf *v,* IntLeaf *mod*)

6.13 Crypto/ElGamal.h File Reference

```
#include "Node.h"
#include "IntLeaf.h"
```

Functions

- IntLeaf PDec (IntLeaf x, IntLeaf c, IntLeaf mod)
- IntLeaf TDec (IntLeaf x, IntLeaf c, IntLeaf mod)
- Node Enc (Node pk, IntLeaf m, IntLeaf s, IntLeaf mod)

6.13.1 Function Documentation

6.13.1.1 Node Enc (Node *pk,* IntLeaf *m,* IntLeaf *s,* IntLeaf *mod*)

6.13.1.2 IntLeaf PDec (IntLeaf *x,* IntLeaf *c,* IntLeaf *mod*)

6.13.1.3 IntLeaf TDec (IntLeaf *x,* IntLeaf *c,* IntLeaf *mod*)

6.14 Crypto/H_SHA.cpp File Reference

```
#include "H_SHA.h"
#include <algorithm>
#include <openssl\sha.h>
```

Functions

- bytevector H_SHA256 (bytevector seed)
- bytevector H_SHA384 (bytevector seed)
- bytevector H_SHA512 (bytevector seed)
- bytevector H_SHA (unsigned char ∗(∗SHA)(const unsigned char ∗d, size_t n, unsigned char ∗md), bytevector seed, unsigned int digest_length)

6.14.1 Function Documentation

6.14.1.1 bytevector H_SHA (unsigned char ∗(∗)(const unsigned char ∗d, size_t n, unsigned char ∗md) *SHA,* bytevector *seed,* unsigned int *digest_length*)

6.14.1.2 bytevector H_SHA256 (bytevector *seed*)

6.14.1.3 bytevector H_SHA384 (bytevector *seed*)

6.14.1.4 bytevector H_SHA512 (bytevector *seed*)

6.15 Crypto/H_SHA.h File Reference

```
#include <vector>
#include "types.h"
```

Functions

- bytevector H_SHA (unsigned char ∗(∗SHA)(const unsigned char ∗d, size_t n, unsigned char ∗md), bytevector seed, unsigned int digest_length)
- bytevector H_SHA256 (bytevector seed)
- bytevector H_SHA384 (bytevector seed)
- bytevector H_SHA512 (bytevector seed)

6.15.1 Function Documentation

6.15.1.1 bytevector H_SHA (unsigned char ∗(∗)(const unsigned char ∗d, size_t n, unsigned char ∗md) *SHA,* bytevector *seed,* unsigned int *digest_length*)

6.15.1.2 bytevector H_SHA256 (bytevector *seed*)

6.15.1.3 bytevector H_SHA384 (bytevector *seed*)

6.15.1.4 bytevector H_SHA512 (bytevector *seed*)

6.16 Crypto/PRG.cpp File Reference

```
#include "PRG.h"
```

6.17 Crypto/PRG.h File Reference

```
#include "Node.h"
#include <string>
#include <vector>
#include <queue>
```

Classes

- class PRG

6.18 Crypto/RandomArray.cpp File Reference

```
#include "RandomArray.h"
```

Functions

- Node RandomArray (Node Gq, unsigned int Nprime, bytevector(∗hash)(bytevector data), bytevector seed, unsigned int Nr)

6.18.1 Function Documentation

6.18.1.1 Node RandomArray (Node *Gq,* unsigned int *Nprime,* bytevector(∗)(bytevector data) *hash,* bytevector *seed,* unsigned int *Nr*)

6.19 Crypto/RandomArray.h File Reference

```
#include "PRG.h"
```

Functions

- Node RandomArray (Node Gq, unsigned int Nprime, bytevector(∗hash)(bytevector data), bytevector seed, unsigned int Nr)

6.19.1 Function Documentation

6.19.1.1 Node RandomArray (Node *Gq,* unsigned int *Nprime,* bytevector(∗)(bytevector data) *hash,* bytevector *seed,* unsigned int *Nr*)

6.20 Crypto/RO.cpp File Reference

```
#include "RO.h"
```

6.21 Crypto/RO.h File Reference

```
#include "Node.h"
#include <string>
#include <vector>
```

Classes

- class RO

6.22 README.md File Reference

6.23 Tests/ByteTreeTests.cpp File Reference

```
#include <gtest/gtest.h>
#include "BaseNode.h"
#include "Node.h"
#include <stdexcept>
```

Functions

- TEST (ByteTreeTests, NodeContructor)
- TEST (ByteTreeTests, ConcatData)
- TEST (ByteTreeTests, Serialize)

6.23.1 Function Documentation

6.23.1.1 TEST (ByteTreeTests , NodeContructor)

6.23.1.2 TEST (ByteTreeTests , ConcatData)

6.23.1.3 TEST (ByteTreeTests , Serialize)

6.24 Tests/IntLeafArithmeticsTests.cpp File Reference

```
#include <gtest/gtest.h>
#include "IntLeaf.h"
```

Functions

- TEST (IntLeafArithmeticTests, MultiplicationPositive)
- TEST (IntLeafArithmeticTests, AdditionPositive)
- TEST (IntLeafArithmeticTests, MultiplicationNegative)
- TEST (IntLeafArithmeticTests, AdditionNegative)

6.24.1 Function Documentation

6.24.1.1 TEST (IntLeafArithmeticTests , MultiplicationPositive)

6.24.1.2 TEST (IntLeafArithmeticTests , AdditionPositive)

6.24.1.3 TEST (IntLeafArithmeticTests , MultiplicationNegative)

6.24.1.4 TEST (IntLeafArithmeticTests , AdditionNegative)

6.25 Tests/NodeArithmeticsTests.cpp File Reference

```
#include <gtest/gtest.h>
#include "IntLeaf.h"
#include "Node.h"
```

Functions

- TEST (NodeArithmeticsTests, AddChildren)
- TEST (NodeArithmeticsTests, VectorAddition)
- TEST (NodeArithmeticsTests, LinkedListNodeAddition)
- TEST (NodeArithmeticsTests, VectorMultiplication)

6.25.1 Function Documentation

6.25.1.1 TEST (NodeArithmeticsTests , AddChildren)

6.25.1.2 TEST (NodeArithmeticsTests , VectorAddition)

6.25.1.3 TEST (NodeArithmeticsTests , LinkedListNodeAddition)

6.25.1.4 TEST (NodeArithmeticsTests , VectorMultiplication)

6.26 Tests/NodeDataInitTests.cpp File Reference

```
#include <gtest/gtest.h>
#include "Node.h"
```

Functions

- TEST (NodeDataInitTests, NodeFromFileTest)
- TEST (NodeDataInitTests, NodeFromFileNameTest)
- TEST (NodeDataInitTests, NodeFromVectorTest)

6.26.1 Function Documentation

6.26.1.1 TEST (NodeDataInitTests , NodeFromFileTest)

6.26.1.2 TEST (NodeDataInitTests , NodeFromFileNameTest)

6.26.1.3 TEST (NodeDataInitTests , NodeFromVectorTest)

6.27 Tests/NodeToStringTests.cpp File Reference

```
#include <gtest/gtest.h>
#include "IntLeaf.h"
#include "Node.h"
```

Functions

- TEST (NodeToStringTests, Vector)
- TEST (NodeToStringTests, PairVector)
- TEST (NodeToStringTests, PairSingle)

6.27.1 Function Documentation

6.27.1.1 TEST (NodeToStringTests , Vector)

6.27.1.2 TEST (NodeToStringTests , PairVector)

6.27.1.3 TEST (NodeToStringTests , PairSingle)

6.28 Tests/PRGTests.cpp File Reference

```
#include <gtest/gtest.h>
#include "H_SHA.h"
#include "PRG.h"
#include <vector>
#include <stdexcept>
#include <algorithm>
#include <iostream>
```

Functions

- TEST (PRGTests, TestVector256)
- TEST (PRGTests, TestVector384)
- TEST (PRGTests, TestVector512)

6.28.1 Function Documentation

6.28.1.1 TEST (PRGTests , TestVector256)

6.28.1.2 TEST (PRGTests , TestVector384)

6.28.1.3 TEST (PRGTests , TestVector512)

6.29 Tests/TestRunner.cpp File Reference

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

Functions

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

6.29.1 Function Documentation

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

6.30 Verifier/DecryptionFactorsVerifier.cpp File Reference

```
#include "DecryptionFactorsVerifier.h"
#include "Node.h"
#include "RO.h"
#include "PRG.h"
#include "H_SHA.h"
#include "ElGamal.h"
#include "Utilities.h"
```

Functions

- bool DecryptionFactorsVerifier (const int j, const proofStruct &ps, const Node &f, const Node &tauDec, const Node &sigmaDec, const Node &w)

6.30.1 Function Documentation

6.30.1.1 **bool DecryptionFactorsVerifier (const int *j,* const proofStruct & *ps,* const Node & *f,* const Node & *tauDec,* const Node & *sigmaDec,* const Node & *w*)**

6.31 Verifier/DecryptionFactorsVerifier.h File Reference

```
#include "Utilities.h"
```

Functions

- bool DecryptionFactorsVerifier (const int j, const proofStruct &ps, const Node &f, const Node &tauDec, const Node &sigmaDec, const Node &w)

6.31.1 Function Documentation

6.31.1.1 **bool DecryptionFactorsVerifier (const int *j,* const proofStruct & *ps,* const Node & *f,* const Node & *tauDec,* const Node & *sigmaDec,* const Node & *w*)**

6.32 Verifier/DecryptionVerifier.cpp File Reference

```
#include "DecryptionVerifier.h"
#include "DecryptionFactorsVerifier.h"
#include "ProofOfShuffle.h"
#include "ElGamal.h"
#include "Node.h"
```

Functions

- bool DecryptionVerifier (const proofStruct &ps, const Node L, const Node m)

6.32.1 Function Documentation

6.32.1.1 **bool DecryptionVerifier (const proofStruct & *ps,* const Node *L,* const Node *m*)**

6.33 Verifier/DecryptionVerifier.h File Reference

```
#include "Node.h"
#include "Utilities.h"
```

Functions

- bool DecryptionVerifier (const proofStruct &ps, const Node L, const Node m)

6.33.1 Function Documentation

6.33.1.1 bool DecryptionVerifier (const proofStruct & *ps*, const Node *L*, const Node *m*)

6.34 Verifier/FileNames.h File Reference

```
#include <string>
```

Variables

- const string FULL_PUBLIC_KEY_FILE = "FullPublicKey.bt"
- const string CIPHERTEXTS_FILE = "Ciphertexts.bt"
- const string PLAINTEXTS_FILE = "Plaintexts.bt"
- const string SHUFFLED_CIPHERTEXTS_FILE = "ShuffledCiphertexts.bt"
- const string PARTIAL_PUBLIC_KEY_FILE_PREFIX = "PublicKey"
- const string PARTIAL_SECRET_KEY_FILE_PREFIX = "SecretKey"
- const string CIPHERTEXTS_FILE_PREFIX = "Ciphertexts"
- const string MAXCIPH_FILE = "maxciph"
- const string FILE_SUFFIX = ".bt"

6.34.1 Variable Documentation

6.34.1.1 const string CIPHERTEXTS_FILE = "Ciphertexts.bt"

6.34.1.2 const string CIPHERTEXTS_FILE_PREFIX = "Ciphertexts"

6.34.1.3 const string FILE_SUFFIX = ".bt"

6.34.1.4 const string FULL_PUBLIC_KEY_FILE = "FullPublicKey.bt"

6.34.1.5 const string MAXCIPH_FILE = "maxciph"

6.34.1.6 const string PARTIAL_PUBLIC_KEY_FILE_PREFIX = "PublicKey"

6.34.1.7 const string PARTIAL_SECRET_KEY_FILE_PREFIX = "SecretKey"

6.34.1.8 const string PLAINTEXTS_FILE = "Plaintexts.bt"

6.34.1.9 const string SHUFFLED_CIPHERTEXTS_FILE = "ShuffledCiphertexts.bt"

6.35 Verifier/KeyVerifier.cpp File Reference

```
#include <vector>
#include <fstream>
#include "KeyVerifier.h"
#include "Node.h"
#include "IntLeaf.h"
#include "FileNames.h"
```

Functions

- bool keyVerifier (int lambda, proofStruct &pfStr)
- bool isPublicKey (const Node &G, const Node &pk)
- bool isPartialPublicKey (const Node &G, const IntLeaf &ppk)
- bool isPartialSecretKey (const Node &G, const IntLeaf &psk)

6.35.1 Function Documentation

6.35.1.1 bool isPartialPublicKey (const Node & *G,* const IntLeaf & *ppk*)

6.35.1.2 bool isPartialSecretKey (const Node & *G,* const IntLeaf & *psk*)

6.35.1.3 bool isPublicKey (const Node & *G,* const Node & *pk*)

6.35.1.4 bool keyVerifier (int *lambda,* proofStruct & *pfStr*)

6.36 Verifier/KeyVerifier.h File Reference

```
#include "Node.h"
#include "Utilities.h"
#include <string>
```

Functions

- bool keyVerifier (int lambda, proofStruct &pfStr)
- bool isPublicKey (const Node &G, const Node &pk)
- bool isPartialPublicKey (const Node &G, const IntLeaf &ppk)
- bool isPartialSecretKey (const Node &G, const IntLeaf &psk)

6.36.1 Function Documentation

6.36.1.1 bool isPartialPublicKey (const Node & *G,* const IntLeaf & *ppk*)

6.36.1.2 bool isPartialSecretKey (const Node & *G,* const IntLeaf & *psk*)

6.36.1.3 bool isPublicKey (const Node & *G,* const Node & *pk*)

6.36.1.4 bool keyVerifier (int *lambda,* proofStruct & *pfStr*)

6.37 Verifier/main.cpp File Reference

```
#include <iostream>
#include <string>
#include "Verifier.h"
```

Functions

- int SetMode (const RunMode new_mode, RunMode &mode)
- int ParseProtinfoDirectory (const int i, char *const argv[], string &protInfo, string &directory)
- int main (int argc, char *argv[])

6.37.1 Function Documentation

6.37.1.1 int main (int *argc,* char * *argv[]*)

6.37.1.2 int ParseProtinfoDirectory (const int *i,* char *const *argv[],* string & *protInfo,* string & *directory*)

6.37.1.3 int SetMode (const RunMode *new_mode,* RunMode & *mode*)

6.38 Verifier/ProofOfShuffle.cpp File Reference

```
#include "ProofOfShuffle.h"
#include "PRG.h"
#include "RO.h"
#include "ElGamal.h"
#include "RandomArray.h"
#include "DataLeaf.h"
#include <cmath>
```

Functions

- bool proofOfShuffle (proofStruct &pfStr, const Node &w, const Node &w_prime, const Node &mu, const Node &tau_pos, const Node &sigma_pos)

6.38.1 Function Documentation

6.38.1.1 bool proofOfShuffle (proofStruct & *pfStr,* const Node & *w,* const Node & *w_prime,* const Node & *mu,* const Node & *tau_pos,* const Node & *sigma_pos*)

6.39 Verifier/ProofOfShuffle.h File Reference

```
#include "Node.h"
#include "Utilities.h"
```

Functions

- bool proofOfShuffle (proofStruct &pfStr, const Node &w, const Node &w_prime, const Node &mu, const Node &tau_pos, const Node &sigma_pos)

6.39.1 Function Documentation

6.39.1.1 bool proofOfShuffle (proofStruct & *pfStr,* const Node & *w,* const Node & *w_prime,* const Node & *mu,* const Node & *tau_pos,* const Node & *sigma_pos*)

6.40 Verifier/ShufflingVerifier.cpp File Reference

```
#include <fstream>
#include <string>
#include "Node.h"
#include "IntLeaf.h"
#include "ShufflingVerifier.h"
#include "ProofOfShuffle.h"
#include "Utilities.h"
#include "FileNames.h"
```

Functions

- bool verifyShuffling (proofStruct &pfStr, int lambda, Node &L0, Node &Llambda, bool posc, bool ccpos)
- bool isListOfCiphertexts (const proofStruct &pfStr, Node &L)

6.40.1 Function Documentation

6.40.1.1 bool isListOfCiphertexts (const proofStruct & *pfStr,* Node & *L*)

6.40.1.2 bool verifyShuffling (proofStruct & *pfStr,* int *lambda,* Node & *L0,* Node & *Llambda,* bool *posc,* bool *ccpos*)

6.41 Verifier/ShufflingVerifier.h File Reference

```
#include <string>
#include "Node.h"
#include "Utilities.h"
```

Functions

- bool verifyShuffling (proofStruct &pfStr, int lambda, Node &L0, Node &Llambda, bool posc, bool ccpos)
- bool isListOfCiphertexts (const proofStruct &pfStr, Node &L)

6.41.1 Function Documentation

6.41.1.1 bool isListOfCiphertexts (const proofStruct & *pfStr,* Node & *L*)

6.41.1.2 bool verifyShuffling (proofStruct & *pfStr,* int *lambda,* Node & *L0,* Node & *Llambda,* bool *posc,* bool *ccpos*)

6.42 Verifier/Utilities.cpp File Reference

```
#include "Utilities.h"
#include <vector>
```

Functions

- bool isElemOfGq (const Node &group, const IntLeaf &elem)
- bool isElemOfZn (const IntLeaf &n, const IntLeaf &elem)

- bool isElemOfMw (const proofStruct &pfStr, const Node &plaintext)
- bool isElemOfCw (const proofStruct &pfStr, const Node &ciphertext)
- bool isPedersenCommitment (const Node &group, const IntLeaf &elem)
- void getGroupFromString (proofStruct &pfStr, std::string str)

6.42.1 Function Documentation

6.42.1.1 void getGroupFromString (proofStruct & *pfStr,* std::string *str*)

6.42.1.2 bool isElemOfCw (const proofStruct & *pfStr,* const Node & *ciphertext*)

6.42.1.3 bool isElemOfGq (const Node & *group,* const IntLeaf & *elem*)

6.42.1.4 bool isElemOfMw (const proofStruct & *pfStr,* const Node & *plaintext*)

6.42.1.5 bool isElemOfZn (const IntLeaf & *n,* const IntLeaf & *elem*)

6.42.1.6 bool isPedersenCommitment (const Node & *group,* const IntLeaf & *elem*)

6.43 Verifier/Utilities.h File Reference

```
#include <string>
#include "Node.h"
#include "IntLeaf.h"
```

Classes

- struct proofStruct

Functions

- bool isElemOfGq (const Node &group, const IntLeaf &elem)
- bool isElemOfZn (const IntLeaf &n, const IntLeaf &elem)
- bool isElemOfCw (const proofStruct &pfStr, const Node &ciphertext)
- bool isElemOfMw (const proofStruct &pfStr, const Node &plaintext)
- bool isPedersenCommitment (const Node &group, const IntLeaf &elem)
- void getGroupFromString (proofStruct &pfStr, std::string str)

Variables

- const IntLeaf BOTTOM = IntLeaf(-1)

6.43.1 Function Documentation

6.43.1.1 void getGroupFromString (proofStruct & *pfStr,* std::string *str*)

6.43.1.2 bool isElemOfCw (const proofStruct & *pfStr,* const Node & *ciphertext*)

6.43.1.3 bool isElemOfGq (const Node & *group,* const IntLeaf & *elem*)

6.43.1.4 bool isElemOfMw (const proofStruct & *pfStr,* const Node & *plaintext*)

6.43.1.5 bool isElemOfZn (const IntLeaf & *n,* const IntLeaf & *elem*)

6.43.1.6 bool isPedersenCommitment (const Node & *group,* const IntLeaf & *elem*)

6.43.2 Variable Documentation

6.43.2.1 const IntLeaf BOTTOM = IntLeaf(-1)

6.44 Verifier/Verifier.cpp File Reference

```
#include "Verifier.h"
#include "Utilities.h"
#include "DecryptionVerifier.h"
#include "ShufflingVerifier.h"
#include "KeyVerifier.h"
#include "FileNames.h"
#include <string>
#include <fstream>
#include <vector>
#include <rapidxml/rapidxml.hpp>
#include "DataLeaf.h"
#include "Node.h"
#include "H_SHA.h"
```

Functions

- int Verifier (string protinfo, string directory, RunMode typeExp, string auxsidExp, int wExp, bool posc, bool ccpos, bool dec)

6.44.1 Function Documentation

6.44.1.1 int Verifier (string *protinfo,* string *directory,* RunMode *typeExp,* string *auxsidExp,* int *wExp,* bool *posc,* bool *ccpos,* bool *dec*)

6.45 Verifier/Verifier.h File Reference

```
#include <string>
```

Enumerations

- enum RunMode {
  NONE, HELP, COMPAT, MIX,
  SHUFFLE, DECRYPT }

Functions

- int Verifier (string protinfo, string directory, RunMode typeExp, string auxsidExp, int wExp, bool posc, bool ccpos, bool dec)

Variables

- const std::string CIPHERTEXT_FILE_PREFIX = "Ciphertexts"

6.45.1 Enumeration Type Documentation

6.45.1.1 enum RunMode

Enumerator

NONE

HELP

COMPAT

MIX

SHUFFLE

DECRYPT

6.45.2 Function Documentation

6.45.2.1 int Verifier (string *protinfo,* string *directory,* RunMode *typeExp,* string *auxsidExp,* int *wExp,* bool *posc,* bool *ccpos,* bool *dec*)

6.45.3 Variable Documentation

6.45.3.1 const std::string CIPHERTEXT_FILE_PREFIX = "Ciphertexts"

Index