# EPiC

Generated by Doxygen 1.7.6.1

Tue Nov 27 2012 17:14:45

# Contents

# Chapter 1

# EPiC documentation

## 1.1 Abstract

In the face of an untrusted cloud infrastructure, outsourced data needs to be protected. Fully homomorphic encryption is one solution that also allows performing operations on outsourced data. However, the involved high overhead of today's fully homomorphic encryption techniques outweigh cloud cost saving advantages, rendering it impractical. We present EPiC, a practical, efficient protocol for the privacy-preserving evaluation of a fundamental operation on data sets: frequency counting. In an IND-CPA encrypted outsourced data set, a cloud user can specify a pattern, and the cloud will count the number of occurrences of this pattern in a completely oblivious manner. A pattern is expressed as a boolean formula on the fields of the records and can specify values counting, range counting, and conjunctions/disjunctions of field values. EPiC's main idea is, first, to reduce the problem of counting to a summation of polynomial evaluations. Second, to efficiently evaluate the summation of polynomial evaluations in a privacy-preserving manner, we extend previous work on the Hidden Modular Group Order assumption and design a new *somewhat homomorphic* encryption scheme. We show how a general pattern, defined by a boolean formula, is arithmetized into a multivariate polynomial over `GF(2)` and used in EPiC. This scheme is highly efficient in our particular counting scenario. Besides a formal analysis where we prove EPiC's privacy, we also present implementation and evaluation results. We specifically target Google's prominent MapReduce paradigm as offered by major cloud providers. Our evaluation performed both locally and in Amazon's public cloud with data sets sizes of up to 1 TByte shows only modest overhead compared to non-private counting, attesting to EPiC's efficiency.

## 1.2 Details

Please read the details `here`.

## 1.3 Implementation

The implementation is done in Java. Please use the navigator above to view the packges, classes and methods.

# Chapter 2

# Namespace Index

## 2.1 Namespace List

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

| | | |
|---|---|---|
| common | Provides classes needed for encryption and polynomial operations | 7 |
| mapred | Provides classes needed for MapReduce jobs | 8 |
| obsolete | Contains obsolete classes | 9 |

# Chapter 3

# Class Index

## 3.1 Class List

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

- common.Benchmark
  Benchmarking the encryption scheme with addition, multiplication and exponentiation . . . . 11
- mapred.BigIntegerWritable
  Provides an *immutable* implementation of a big integer for use in Hadoop framework . . . . 12
- obsolete.BigVector . . . . 15
- common.Cipher
  Provides encryption and decryption operations . . . . 18
- common.ClientRequest
  Computes the user's request as a set of encrypted coefficients corresponding to the queried indicator polynomial . . . . 20
- mapred.Count
  EPiC MapReduce main class . . . . 22
- mapred.CustomRecordReader
  Provides an implementation of a common record reader for all MapReduce jobs in the distributed application . . . . 25
- common.GenPrime
  Generates prime `p` and prime `q` for the encryption scheme . . . . 26
- common.GetAnswer
  Provides a tool to obtain the plain-text count value from the answer received from the cloud . . . . 27
- common.LocalCount . . . . 28
- mapred.MapRedEpic
  Implementation of EPiC's approach . . . . 29
- mapred.MapRedEpicReducerEvaluate
  Implementation of EPiC's approach with a slight difference . . . . 30
- mapred.MapRedNotSendCoeff
  This is an older implementation of EPiC . . . . 31

# Chapter 4

# Namespace Documentation

## 4.1 Package common

Provides classes needed for encryption and polynomial operations.

### Classes

- class Benchmark

  *Benchmarking the encryption scheme with addition, multiplication and exponentiation.*
- class Cipher

  *Provides encryption and decryption operations.*
- class ClientRequest

  *Computes the user's request as a set of encrypted coefficients corresponding to the queried indicator polynomial.*
- class GenPrime

  *Generates prime* $p$ *and prime* $q$ *for the encryption scheme.*
- class GetAnswer

  *Provides a tool to obtain the plain-text count value from the answer received from the cloud.*
- class LocalCount
- class MultiArray$<$ T $>$
- class MultiFieldKey
- class Parameters

  *Provides tools for handling the parameters used for the encryption scheme.*
- class Polynomial

  *Provides an implementation for a univariate polynomial* `P(x)`.
- class Producer

  *Implementation of a local data generator.*
- class Record

  *Definition of a record used by the application.*

- class Statistics

  *Provides a tool for collecting statistics about the values of fields in the data set.*

### 4.1.1 Detailed Description

Provides classes needed for encryption and polynomial operations.

**Encryption**
- **KeyGen**: primes $p$ and $q$ are generated in GenPrime and security parameters and other parameters are handled in Parameters.
- **Enc**, **Dec**: encryption and decryption are implemented in Cipher.
- The encryption scheme is benchmarked by Benchmark.

**Operations on polynomials**
- An implementation of a univariate polynomial is provided by Polynomial.
- User's request is generated by ClientRequest.
- Processing of cloud's answer is handled at the user side by GetAnswer. For MapRedNotSendCoeff, use LocalCount instead.

**Data manipulation**
- A data set contains a number of records. Each record has multiple fields. The storing format of each record is defined in Record.
- Multiple fields in a record are handled in MapReduce framework by MultiFieldKey and MultiArray.
- To generate data, see Producer. While data is generate, statistics are collected by Statistics.

**Author**

vohuudtr

## 4.2 Package mapred

Provides classes needed for MapReduce jobs.

### Classes

- class BigIntegerWritable

  *Provides an immutable implementation of a big integer for use in Hadoop framework.*
- class Count

  *EPiC MapReduce main class.*
- class CustomRecordReader

  *Provides an implementation of a common record reader for all MapReduce jobs in the distributed application.*
- class MapRedEpic

  *Implementation of EPiC's approach.*
- class MapRedEpicReducerEvaluate

  *Implementation of EPiC's approach with a slight difference.*
- class MapRedNotSendCoeff

  *This is an older implementation of EPiC.*
- class MapRedPlainCountAll

  *This is similar to MapRedPlainCountOne, but supports counting many values at once.*
- class MapRedPlainCountOne

  *This is an illustrating implementation of counting based on unencrypted fields.*
- class Producer

  *This is a MapReduce job used for generating a large data of set and storing in the HDFS.*
- class RecordInputFormat

  *Provides customized input format for MapReduce counting jobs.*

### 4.2.1 Detailed Description

Provides classes needed for MapReduce jobs. The main executable class is Count, which receives command-line parameters and executes the MapReduce counting job.

Different counting approaches are implemented in classes prefixed with `MapRed`.

The Producer class is an implementation of a MapReduce job used to generate a large data set in parallel to reduce the generating time.

## 4.3 Package obsolete

Contains obsolete classes.

### Classes

- class BigVector
- class PolyMatrix

### 4.3.1 Detailed Description

Contains obsolete classes.

**Author**

vohuudtr

# Chapter 5

# Class Documentation

## 5.1 common.Benchmark Class Reference

Benchmarking the encryption scheme with addition, multiplication and exponentiation.

### Static Public Member Functions

- static void main (String[] args)

### 5.1.1 Detailed Description

Benchmarking the encryption scheme with addition, multiplication and exponentiation.

**Author**

vohuudtr

### 5.1.2 Member Function Documentation

#### 5.1.2.1 static void **common.Benchmark.main** ( String[] *args* ) `[inline, static]`

**Parameters**

| | |
|---|---|
| *args* | |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Benchmark.java

## 5.2 mapred.BigIntegerWritable Class Reference

Provides an *immutable* implementation of a big integer for use in Hadoop framework.

Inherits Writable.

### Public Member Functions

- BigIntegerWritable ()

  *Default constructor without initialization.*
- BigIntegerWritable (int n)

  *Constructs a BigIntegerWritable object from an integer.*
- BigIntegerWritable (BigInteger n)

  *Constructs a BigIntegerWritable object from a Java BigInteger object.*
- BigIntegerWritable (DataInput in) throws IOException

  *Constructs a BigIntegerWritable object from the input stream.*
- void readFields (DataInput in) throws IOException

  *Initializes the object by reading data from the input stream.*
- void write (DataOutput out) throws IOException

  *Writes the object into the specified output stream.*
- int getSize ()

  *Returns the total size in bytes representing the value of the object.*
- BigIntegerWritable add (BigIntegerWritable a)

  *Returns a new BigIntegerWritable object with value equal to the sum of this object's value and another object's value.*
- BigIntegerWritable multiply (BigIntegerWritable a)

  *Returns a new BigIntegerWritable object with value equal to the product of this object's value and another object's value.*
- String toString ()

  *Returns a string representing this object.*

### 5.2.1 Detailed Description

Provides an *immutable* implementation of a big integer for use in Hadoop framework.

Similarly to the Java BigInteger class, this class supports two basic mathematical operations: **addition** and **multiplication**. Besides, it also implements the Writable interface to support **reading** and **writing** operations in Hadoop framework.

**Author**

vohuudtr

## 5.2.2 Constructor & Destructor Documentation

### 5.2.2.1 mapred.BigIntegerWritable.BigIntegerWritable ( ) `[inline]`

Default constructor without initialization.

This constructor is needed for the iterator used in the reduce method of `MapRed`-classes. Therefore, one should *manually* initialize the value of this object after using this constructor.

### 5.2.2.2 mapred.BigIntegerWritable.BigIntegerWritable ( int *n* ) `[inline]`

Constructs a BigIntegerWritable object from an integer.

**Parameters**

| | |
|---|---|
| *n* | initialized value for the constructed object. |

### 5.2.2.3 mapred.BigIntegerWritable.BigIntegerWritable ( BigInteger *n* ) `[inline]`

Constructs a BigIntegerWritable object from a Java BigInteger object.

**Parameters**

| | |
|---|---|
| *n* | initialized value for the constructed object. |

### 5.2.2.4 mapred.BigIntegerWritable.BigIntegerWritable ( DataInput *in* ) throws IOException `[inline]`

Constructs a BigIntegerWritable object from the input stream.

This constructor is a short-hand for initializing the object with the default constructor and calling readFields(DataInput) to read the value from the input stream.

**Parameters**

| | |
|---|---|
| *in* | input stream |

**Exceptions**

| | |
|---|---|
| *IOException* | if IO errors occur. |

**See also**

readFields(DataInput) for storing format of the object in the stream.

### 5.2.3 Member Function Documentation

#### 5.2.3.1 BigIntegerWritable mapred.BigIntegerWritable.add ( BigIntegerWritable *a* ) [inline]

Returns a new BigIntegerWritable object with value equal to the sum of this object's value and another object's value.

**Parameters**

| | |
|---|---|
| *a* | another object. |

**Returns**

result of the addition.

#### 5.2.3.2 int mapred.BigIntegerWritable.getSize ( ) [inline]

Returns the total size in bytes representing the value of the object.

**Returns**

total size in bytes.

#### 5.2.3.3 BigIntegerWritable mapred.BigIntegerWritable.multiply ( BigIntegerWritable *a* ) [inline]

Returns a new BigIntegerWritable object with value equal to the product of this object's value and another object's value.

**Parameters**

| | |
|---|---|
| *a* | another object. |

**Returns**

result of the multiplication.

#### 5.2.3.4 void mapred.BigIntegerWritable.readFields ( DataInput *in* ) throws IOException [inline]

Initializes the object by reading data from the input stream.

The object is stored in a stream as an *ordered sequence* of bytes:

- First 4 bytes represents an integer which specifies the number of bytes used for storing the "big-integer" value of this object.

- The value of the object as an array of bytes with the length specified in the above 4 bytes. The bytes storing order depends on the Java BigInteger implementation.

**Parameters**

| | |
|---|---|
| *in* | input stream. |

**Exceptions**

| | |
|---|---|
| *IOException* | if IO errors occur. |

#### 5.2.3.5 void **mapred.BigIntegerWritable.write** ( DataOutput *out* ) throws IOException `[inline]`

Writes the object into the specified output stream.

The sequence of bytes representing the object is in the format described in readFields(DataInput).

**Parameters**

| | |
|---|---|
| *out* | output stream. |

**Exceptions**

| | |
|---|---|
| *IOException* | if IO errors occur. |

**See also**

readFields(DataInput) for storing format of the object in the stream.

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/BigIntegerWritable.java

## 5.3 obsolete.BigVector Class Reference

### Public Member Functions

- BigVector (int size)

  *Constructing a big vector with given size and initializing all elements to 0.*
- BigVector (BigVector v)

  *Constructing a big vector from a given big-vector.*
- BigVector (BigInteger[] values)

*Constructing a big vector with given big-integer values.*
- BigVector (int[ ] values)

  *Constructing a big vector with given small-integer values.*
- BigVector (Polynomial p)

  *Constructing a big vector with coefficients of given polynomial as initial values.*
- BigInteger get (int which)

  *Return value of the element at "which" position.*
- void set (int which, BigInteger value)

  *Set value for the element at "which" position.*
- BigInteger multiply (BigVector v)

  *Perform scalar product between two big vectors and return the result.*
- String **toString** ()

## 5.3.1 Constructor & Destructor Documentation

### 5.3.1.1 obsolete.BigVector.BigVector ( int *size* ) `[inline]`

Constructing a big vector with given size and initializing all elements to 0.

**Parameters**

| | |
|---|---|
| *size* | size of the vector. |

### 5.3.1.2 obsolete.BigVector.BigVector ( BigVector *v* ) `[inline]`

Constructing a big vector from a given big-vector.

**Parameters**

| | |
|---|---|
| *v* | another big-vector. |

### 5.3.1.3 obsolete.BigVector.BigVector ( BigInteger[ ] *values* ) `[inline]`

Constructing a big vector with given big-integer values.

**Parameters**

| | |
|---|---|
| *values* | initial big-integer values. |

### 5.3.1.4 obsolete.BigVector.BigVector ( int[ ] *values* ) `[inline]`

Constructing a big vector with given small-integer values.

**Parameters**

| | |
|---:|---|
| *values* | initial small-integer values. |

#### 5.3.1.5 obsolete.BigVector.BigVector ( Polynomial *p* ) `[inline]`

Constructing a big vector with coefficients of given polynomial as initial values.

**Parameters**

| | |
|---:|---|
| *p* | polynomial whose coefficients used as initial values for the vector. |

### 5.3.2 Member Function Documentation

#### 5.3.2.1 BigInteger obsolete.BigVector.get ( int *which* ) `[inline]`

Return value of the element at "which" position.

**Parameters**

| | |
|---:|---|
| *which* | position of the requested element. |

**Returns**

value of the requested element.

#### 5.3.2.2 BigInteger obsolete.BigVector.multiply ( BigVector *v* ) `[inline]`

Perform scalar product between two big vectors and return the result.

**Parameters**

| | |
|---:|---|
| *v* | the vector to be multiplied. |

**Returns**

scalar product of two vectors.

#### 5.3.2.3 void obsolete.BigVector.set ( int *which,* BigInteger *value* ) `[inline]`

Set value for the element at "which" position.

**Parameters**

| | |
|---:|---|
| *which* | position to be set. |
| *value* | value to be set. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/obsolete/BigVector.java

## 5.4 common.Cipher Class Reference

Provides encryption and decryption operations.

### Static Public Member Functions

- static void initialize (String paramfile)

  *Initializes the cipher with parameters in a given file.*
- static void initialize (InputStream is)

  *Initializes the cipher with parameters in a stream.*
- static BigInteger encrypt (BigInteger x)

  *Encrypts an integer `x` to `E(x)`.*
- static BigInteger decrypt (BigInteger c)

  *Decrypts an encrypted value to the original integer value.*
- static BigInteger decrypt (BigInteger c, int degree)

  *Decrypts an encrypted value of a specified degree of `b` to the original integer value.*

### 5.4.1 Detailed Description

Provides encryption and decryption operations.

The used encryption scheme is

$y = E(x) = b \cdot (r \cdot q + x) \bmod p$

The decryption is done by

$x = D(y) = b^{-1} \cdot y \bmod p \bmod q$

If an encrypted value is known to contain `b` of a degree `k` greater than 1, the decryption is instead done by $x = D(y,k) = b^{-k} \cdot y \bmod p \bmod q$

Before using, the Cipher object needs to be initialized with either a parameters file (see initialize(String)) or a stream (see initialize(InputStream)) containing the parameters.

**Author**

vohuudtr

### 5.4.2 Member Function Documentation

#### 5.4.2.1 static BigInteger common.Cipher.decrypt ( BigInteger *c* ) `[inline, static]`

Decrypts an encrypted value to the original integer value.

**Parameters**

| | |
|---|---|
| *c* | encrypted value. |

**Returns**

plain-text integer.

#### 5.4.2.2 static BigInteger common.Cipher.decrypt ( BigInteger *c*, int *degree* ) `[inline, static]`

Decrypts an encrypted value of a specified degree of `b` to the original integer value.

**Parameters**

| | |
|---|---|
| *c* | encrypted value. |
| *degree* | known degree of `b` in the encrypted value. |

**Returns**

plain-text integer.

#### 5.4.2.3 static BigInteger common.Cipher.encrypt ( BigInteger *x* ) `[inline, static]`

Encrypts an integer `x` to `E(x).`

**Parameters**

| | |
|---|---|
| *x* | integer to be encrypted. |

**Returns**

encrypted integer.

#### 5.4.2.4 static void common.Cipher.initialize ( String *paramfile* ) `[inline, static]`

Initializes the cipher with parameters in a given file.

**Parameters**

| | |
|---|---|
| *paramfile* | parameters filename. |

#### 5.4.2.5 static void **common.Cipher.initialize** ( InputStream *is* ) `[inline, static]`

Initializes the cipher with parameters in a stream.

**Parameters**

| | |
|---|---|
| *is* | stream containing the parameters. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Cipher.java

## 5.5 common.ClientRequest Class Reference

Computes the user's request as a set of encrypted coefficients corresponding to the queried indicator polynomial.

### Static Public Member Functions

- static void requestEpic (MultiFieldKey key, DataOutput out)

  *Makes request for MapRedEpic.*
- static void requestPlain (MultiFieldKey key, DataOutput out)

  *Makes request for MapRedPlainCountOne.*
- static void requestNotSendCoeff (MultiFieldKey key, DataOutput out)

  *Makes request for MapRedNotSendCoeff.*
- static void main (String[ ] args)

### 5.5.1 Detailed Description

Computes the user's request as a set of encrypted coefficients corresponding to the queried indicator polynomial.

The current implementation supports computing the query only for a specified value over given fields. General boolean expressions are not supported, thus should be done manually.

Usage:

```
java common.ClientRequest <mapred> <key> <paramfile>
<requestfile>
```

**mapred epic** Make request for MapRedEpic.

**plain** Make request for MapRedPlainCountOne.

**notsendcoeff** Make request for MapRedNotSendCoeff.

**key** Value to be counted.

**paramfile** Path to the parameters file (see Parameters).

**requestfile** Path to the request to be created.

**Author**

vohuudtr

### 5.5.2 Member Function Documentation

#### 5.5.2.1 static void **common.ClientRequest.main** ( String[] *args* ) `[inline, static]`

**Parameters**

| | |
|---|---|
| *args* | |

#### 5.5.2.2 static void **common.ClientRequest.requestEpic** ( **MultiFieldKey** *key*, DataOutput *out* ) `[inline, static]`

Makes request for MapRedEpic.

**Parameters**

| | |
|---|---|
| *key* | value to be counted. |
| *out* | output stream containing the created request. |

#### 5.5.2.3 static void **common.ClientRequest.requestNotSendCoeff** ( **MultiFieldKey** *key*, DataOutput *out* ) `[inline, static]`

Makes request for MapRedNotSendCoeff.

This is actually a **fake** method as we do not send any coefficients when using MapRedNotSendCoeff. Processing of the query in this case is actually done by LocalCount, which handles the request and answer locally. So the arguments provided to this method can be `null`.

**Parameters**

| | |
|---|---|
| *key* | value to be counted, can be `null`. |
| *out* | output stream containing the created request. The output stream can be `null`. |

#### 5.5.2.4 static void **common.ClientRequest.requestPlain** ( **MultiFieldKey** *key*, DataOutput *out* ) `[inline, static]`

Makes request for MapRedPlainCountOne.

**Parameters**

| | |
|---|---|
| *key* | value to be counted. |
| *out* | output stream containing the created request. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/ClientRequest.java

## 5.6 mapred.Count Class Reference

EPiC MapReduce main class.

Inherits Configured, and Tool.

### Public Member Functions

- int run (String[ ] args) throws Exception

  *Run the job.*

### Static Public Member Functions

- static void main (String[ ] args) throws Exception

  *Entry point of the class.*

### Static Package Functions

- static DataInput getInput (Path[ ] files, String name)

  *Returns the data input interface for a file with specified patterns among given file paths.*

### 5.6.1 Detailed Description

EPiC MapReduce main class.

Based on provided parameters via command line, an appropriate MapReduce class is executed. Currently, there are two implementation of counting job that can be called via Count:

**MapRedEpic** EPiC approach - counting on encrypted fields.

**MapRedPlainCountOne** Plain-text counting.

Usage of Count via command-line:

```
hadoop jar <JARFILE> mapred.Count [options] <input> <output>
```

**JARFILE** The JAR file containing EPiC.

**input** A HDFS path to the directory containing the input data.

*Note:* Only input files in the specified directory which start with "data" are read.

**output** A HDFS path to the directory containing the results.

*Note:* Existing output will be *automatically* removed when Count is started.

The options given to Hadoop must be prefixed by "-D". The following options are supported:

**paramfile** Name of the encryption parameters file. Path to the parameters file must be relative to this class inside the the JAR file.

Example: `-Dparamfile=params.txt`

**request** A HDFS path to the user's request file containing the counting query. The specified path is relative to the **input** directory.

Example: `-Drequest=request`

**mapred** Specifies which MapReduce approach to be executed. Currently the following values are supported:

**epic** Calling MapRedEpic to execute the query, which applies a variant version of the approach presented in the EPiC paper. Precisely, based on the provided `request`, which contains the encrypted coefficients of the queried indicator polynomial, the Mappers evaluate the indicator polynomial for each record by multiplying the monomials with the coefficients before adding them together. In the last step at the Reducer, those results from Mappers (now considered as the value of the indicator polynomial evaluated for the corresponding subsets) are added together to obtain the final results and return to the user.

The approach presented in the EPiC paper is implemented in MapRedEpicReducerEvaluate, in which the Mappers compute the monomials without multiplying with the coefficients. At the final step, the Reducer adds those results from the Mappers together and then multiplies with the given coefficients to yield the final results.

An older approach of EPiC (see MapRedNotSendCoeff) is to keep the coefficients at the user side. The Mappers and Reducers only need to compute the monomials and add them together, then return to the user, who will be responsible to multiply the results with the precomputed coefficients to obtain the counting result. This, however, requires much more communication for downloading the results, therefore, is impractical.

**plain** Calling MapRedPlainCountOne to count on plain-text values of the multiple countable fields. This implementation does not support range counting, boolean expressions, etc. Another illustration of plain-text counting is implemented in MapRedPlainCountAll which counts all possible values in one MapReduce job.

Example: `-Dmapred=epic`

**Author**

vohuudtr

## 5.6.2 Member Function Documentation

### 5.6.2.1 static DataInput **mapred.Count.getInput** ( Path[] *files*, String *name* ) `[inline, static, package]`

Returns the data input interface for a file with specified patterns among given file paths.

This method should be used inside the package only.

**Parameters**

| | |
|---|---|
| *files* | set of files. |
| *name* | pattern that ends the files. |

**Returns**

the data input interface corresponding to the matched file.

### 5.6.2.2 static void **mapred.Count.main** ( String[] *args* ) throws Exception `[inline, static]`

Entry point of the class.

**Parameters**

| | |
|---|---|
| *args* | command-line arguments provided to the class. |

**Exceptions**

| | |
|---|---|
| *Exception* | if errors occur. |

### 5.6.2.3 int **mapred.Count.run** ( String[] *args* ) throws Exception `[inline]`

Run the job.

**Parameters**

| | |
|---|---|
| *args* | argument list for the running job. |

**Exceptions**

| | |
|---|---|
| *Exception* | if errors occur. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/Count.java

## 5.7 mapred.CustomRecordReader Class Reference

Provides an implementation of a common record reader for all MapReduce jobs in the distributed application.

Inherits RecordReader< LongWritable, Record >.

### Public Member Functions

- **CustomRecordReader** (JobConf job, FileSplit split) throws IOException
- void **close** () throws IOException
- LongWritable **createKey** ()
- Record **createValue** ()
- long **getPos** () throws IOException
- float **getProgress** () throws IOException
- boolean **next** (LongWritable key, Record record) throws IOException

### 5.7.1 Detailed Description

Provides an implementation of a common record reader for all MapReduce jobs in the distributed application.

Each record has a fixed size, defined in Record.

**Author**

vohuudtr

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/CustomRecordReader.java

## 5.8 common.GenPrime Class Reference

Generates prime `p` and prime `q` for the encryption scheme.

### Static Public Member Functions

- static int calculateSizeQ ()

  *Calculates the required size in bits of prime* $q$*.*
- static int calculateSizeP (int size_q)

  *Calculates the required size in bits of prime* $p$*.*
- static void generatePrime ()

  *Generates prime* $p$ *and* $q$*.*

### 5.8.1 Detailed Description

Generates prime `p` and prime `q` for the encryption scheme.

The primes are generated based on the provided security parameters in Parameters.

**Author**

vohuudtr

### 5.8.2 Member Function Documentation

#### 5.8.2.1 static int **common.GenPrime.calculateSizeP** ( int *size_q* ) `[inline, static]`

Calculates the required size in bits of prime `p`.

This calculation requires the calculation of `q` to be done first.

**Parameters**

| | |
|---|---|
| *size_q* | size of `q`. |

**Returns**

number of bits of prime `p`.

#### 5.8.2.2 static int **common.GenPrime.calculateSizeQ** ( ) `[inline, static]`

Calculates the required size in bits of prime `q`.

**Returns**

number of bits of prime `q`.

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/GenPrime.java

## 5.9 common.GetAnswer Class Reference

Provides a tool to obtain the plain-text count value from the answer received from the cloud.

### Static Public Member Functions

- static void parseAnswerEpic (String answer)

  *Decrypts the encrypted answer to the plain-text count value.*
- static void parseAnswerPlain (String answer)

  *In plain-text MapReduce cases, the received answer is also the count value.*
- static void main (String[] args)

### 5.9.1 Detailed Description

Provides a tool to obtain the plain-text count value from the answer received from the cloud.

The answer is simply decrypted using the known degree of `b`.

Usage:

```
java common.GetAnswer <mapred> <paramfile> <answerfile>
```

**mapred** **epic** Gets answer received from MapRedEpic.

**plain** Gets answer received from MapRedPlainCountOne.

**paramfile** Path to the parameters file (see Parameters).

**answerfile** Path to the answer file.

**Author**

vohuudtr

### 5.9.2 Member Function Documentation

#### 5.9.2.1 static void **common.GetAnswer.main** ( String[] *args* ) `[inline, static]`

**Parameters**

| | |
|---|---|
| *args* | |

#### 5.9.2.2 static void **common.GetAnswer.parseAnswerEpic** ( String *answer* ) `[inline, static]`

Decrypts the encrypted answer to the plain-text count value.

The plain-text value is printed to the standard output.

**Parameters**

| | |
|---|---|
| *answer* | encrypted answer as a big-integer in Java String format. |

#### 5.9.2.3 static void **common.GetAnswer.parseAnswerPlain** ( String *answer* ) `[inline, static]`

In plain-text MapReduce cases, the received answer is also the count value.

The answer is, therefore, printed directly to the standard output.

**Parameters**

| | |
|---|---|
| *answer* | plain-text answer as a big-integer in Java String format. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/GetAnswer.java

## 5.10 common.LocalCount Class Reference

### Public Member Functions

- **LocalCount** (String paramfile, String datafile, String origstatfile, String resultfile)
- void count ()

  *Read encrypted big integers from file, decrypt them, and count number of each of them.*
- int[ ] **domain** (int size)
- int **computeBinary** (int ind, MultiArray< BigInteger > sum)
- BigInteger **compute** (int ind, MultiArray< BigInteger > sum)
- void **computeCoeffs** (Polynomial[ ] f, MultiArray< BigInteger > coeffs, MultiFieldKey runningKey, BigInteger runningCoeff, int runningField)
- void **readSum** (MultiArray< BigInteger > sum) throws IOException
- void **readOrigStat** (MultiArray< BigInteger > origsum) throws IOException

## Static Public Member Functions

- static void **usage** ()
- static void main (String[] args)

## Package Attributes

- String **origstatfile**
- String **resultfile**

### 5.10.1 Member Function Documentation

#### 5.10.1.1 void **common.LocalCount.count** ( ) `[inline]`

Read encrypted big integers from file, decrypt them, and count number of each of them.

**Exceptions**

| | |
|---|---|
| *IOException* | IO exception. |

#### 5.10.1.2 static void **common.LocalCount.main** ( String[] *args* ) `[inline, static]`

**Parameters**

| | |
|---|---|
| *args* | |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/LocalCount.java

# 5.11 mapred.MapRedEpic Class Reference

Implementation of EPiC's approach.

## Classes

- class **CountMapper**
- class **CountReducer**

## Static Public Member Functions

- static void setup (JobConf conf)

  *Initializes the class based on provided configuration parameters.*

### 5.11.1 Detailed Description

Implementation of EPiC's approach.

The Mappers evaluate the indicator polynomial for the Mappers' input split by computing the monomials for each record and multiplying them with the encrypted coefficients of the given polynomial in the user's request. Results for each record in the subset are added together and sent to the Reducer. The Reducer simply adds the results from Mappers together and obtain the final results for the whole data set.

This implementation supports all kinds of counting (conjunctive, disjunctive, range, G-F(2) arithmetized).

**Author**

vohuudtr

### 5.11.2 Member Function Documentation

#### 5.11.2.1 static void **mapred.MapRedEpic.setup** ( JobConf *conf* ) `[inline, static]`

Initializes the class based on provided configuration parameters.

The initialization comprises setting job's name, assigning Mapper and Reducer class as well as their input and output key-value classes.

**Parameters**

| | |
|---|---|
| *conf* | configuration parameters. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/MapRedEpic.java

## 5.12 mapred.MapRedEpicReducerEvaluate Class Reference

Implementation of EPiC's approach with a slight difference.

### Classes

- class **CountMapper**
- class **CountReducer**

### Static Public Member Functions

- static void setup (JobConf conf)

  *Initializes the class based on provided configuration parameters.*

### 5.12.1 Detailed Description

Implementation of EPiC's approach with a slight difference.

The difference from MapRedEpic is that the evaluation of the indicator polynomial is switched from the Mappers to the Reducer. Therefore, instead of sending only the final results corresponding to the subsets, all monomials are sent from Mappers to Reducer, which increases the communication among nodes. This is a "naive" implementation of EPiC.

This implementation supports all kinds of counting (conjunctive, disjunctive, range, G-F(2) arithmetized).

**Author**

vohuudtr

### 5.12.2 Member Function Documentation

#### 5.12.2.1 static void **mapred.MapRedEpicReducerEvaluate.setup** ( JobConf *conf* ) `[inline, static]`

Initializes the class based on provided configuration parameters.

The initialization comprises setting job's name, assigning Mapper and Reducer class as well as their input and output key-value classes.

**Parameters**

| | |
|---|---|
| *conf* | configuration parameters. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/MapRedEpicReducerEvaluate.java

## 5.13 mapred.MapRedNotSendCoeff Class Reference

This is an older implementation of EPiC.

### Classes

- class **CountMapper**
- class **CountReducer**

### Static Public Member Functions

- static void setup (JobConf conf)

*Initializes the class based on provided configuration parameters.*

### 5.13.1 Detailed Description

This is an older implementation of EPiC.

The Mappers and Reducer only compute the monomials and return back to the user. The user is responsible for reconstructing the count value by multiplying the obtained results with the precomputed coefficients.

**Author**

vohuudtr

### 5.13.2 Member Function Documentation

#### 5.13.2.1 static void **mapred.MapRedNotSendCoeff.setup** ( JobConf *conf* ) `[inline, static]`

Initializes the class based on provided configuration parameters.

The initialization comprises setting job's name, assigning Mapper and Reducer class as well as their input and output key-value classes.

**Parameters**

| | |
|---|---|
| *conf* | configuration parameters. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/MapRedNotSendCoeff.java

## 5.14 mapred.MapRedPlainCountAll Class Reference

This is similar to MapRedPlainCountOne, but supports counting many values at once.

### Classes

- class **CountMapper**
- class **CountReducer**

### Static Public Member Functions

- static void setup (JobConf conf)

  *Initializes the class based on provided configuration parameters.*

### 5.14.1 Detailed Description

This is similar to MapRedPlainCountOne, but supports counting many values at once.

Still, it does not support disjunctive, arbitrary boolean expression, GF(2) arithmetized counting.

**Author**

vohuudtr

### 5.14.2 Member Function Documentation

#### 5.14.2.1 static void **mapred.MapRedPlainCountAll.setup** ( JobConf *conf* ) `[inline, static]`

Initializes the class based on provided configuration parameters.

The initialization comprises setting job's name, assigning Mapper and Reducer class as well as their input and output key-value classes.

**Parameters**

| | |
|---|---|
| *conf* | configuration parameters. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/MapRedPlainCountAll.java

## 5.15 mapred.MapRedPlainCountOne Class Reference

This is an illustrating implementation of counting based on unencrypted fields.

### Classes

- class **CountMapper**
- class **CountReducer**

### Static Public Member Functions

- static void setup (JobConf conf)

  *Initializes the class based on provided configuration parameters.*

### 5.15.1 Detailed Description

This is an illustrating implementation of counting based on unencrypted fields.

This implementation supports single field counting and multiple conjuctive counting only. Disjunctive counting (and therefore, arbitrary boolean expressions) is not supported.

**Author**

vohuudtr

### 5.15.2 Member Function Documentation

#### 5.15.2.1 static void **mapred.MapRedPlainCountOne.setup** ( JobConf *conf* ) `[inline, static]`

Initializes the class based on provided configuration parameters.

The initialization comprises setting job's name, assigning Mapper and Reducer class as well as their input and output key-value classes.

**Parameters**

| | |
|---|---|
| *conf* | configuration parameters. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/MapRedPlainCountOne.java

## 5.16 common.MultiArray< T > Class Reference

### Public Member Functions

- T **get** (MultiFieldKey key)
- T **get** (int ind)
- void **put** (MultiFieldKey key, T value)
- void **put** (int ind, T value)
- T[] **getAll** ()
- int **getSize** ()

### Static Public Member Functions

- static MultiArray < BigIntegerWritable > **createBigIntegerWritableArray** (int size)
- static MultiArray< BigInteger > **createBigIntegerArray** (int size)
- static MultiArray< Integer > **createIntegerArray** (int size)

- static MultiArray < BigIntegerWritable > **readMultiArrayBigIntegerWritable** (DataInput in) throws IOException
- static void **write** (MultiArray< BigIntegerWritable > a, DataOutput out) throws IOException
- static int **getIndex** (MultiFieldKey key)
- static MultiFieldKey **getMultiFieldKey** (int ind)

### 5.16.1 Detailed Description

**Author**

vohuudtr

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/MultiArray.java

## 5.17 common.MultiFieldKey Class Reference

Inherits Writable, and WritableComparable< MultiFieldKey >.

### Public Member Functions

- **MultiFieldKey** (int numFields)
- **MultiFieldKey** (DataInput in) throws IOException
- int **weight** ()
- boolean **equals** (Object k)
- int **hashCode** ()
- String **toString** ()
- void **readFields** (DataInput in) throws IOException
- void **write** (DataOutput out) throws IOException
- int **compareTo** (MultiFieldKey key)
- MultiFieldKey **clone** ()

### Public Attributes

- int[ ] **element**

### 5.17.1 Detailed Description

**Author**

vohuudtr

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/MultiFieldKey.java

## 5.18 common.Parameters Class Reference

Provides tools for handling the parameters used for the encryption scheme.

### Static Public Member Functions

- static BigInteger getP ()

  *Returns secret prime* $p$*.*
- static void setP (BigInteger newp)

  *Sets new secret prime* $p$*.*
- static BigInteger getB ()

  *Returns secret* $b$*.*
- static void setB (BigInteger newb)

  *Sets the new value for* $b$*.*
- static BigInteger getQ ()

  *Returns prime* $q$*.*
- static void setQ (BigInteger newq)

  *Sets new prime* $q$*.*
- static BigInteger getN ()

  *Returns upperbound of the number of records.*
- static int getS1 ()

  *Returns security parameter* $s1$*.*
- static int getS2 ()

  *Returns security parameter* $s2$*.*
- static int getSizeD (int field)

  *Returns domain size of a field.*
- static int getMaxSizeD ()

  *Returns the maximum domain size over all the countable fields.*
- static int getNumFields ()

  *Returns number of countable fields.*
- static int getTotalSizeD ()

  *Returns the total domain size which is the product of domain size of all fields.*
- static void writeConfig (String filename)

  *Writes parameters to a configuration file.*
- static void writeConfig (OutputStream os)

  *Writes parameters to an output stream.*
- static void readConfig (String filename)

  *Reads parameters from a configuration file.*
- static void readConfig (InputStream is)

  *Reads parameters from an input stream.*
- static void main (String[] args)

### Static Package Functions

- static void **tic** ()
- static void **toc** ()

## 5.18.1 Detailed Description

Provides tools for handling the parameters used for the encryption scheme.

The following parameters are controlled:

**Secret key** **p** secret prime `p` in the encryption scheme.

  **b** secret random `b` in the encryption scheme.

**Public parameters** **q** prime `q` in the encryption scheme.

  **s1** security parameter `s1` in the encryption scheme.

  **s2** maximum size in bits of the random `r` in the encryption scheme.

  **n** upperbound of the number of records in the data set.

  **numFields** number of countable fields in each record.

  > *Note:* **numFields** is the actual number of countable fields in a record, while Record#getNumFields() returns the number of fields contained in a record, which includes both encrypted and plain-text fields, so Record#getNumFields() is equal to `2*numFields` (see Record).

  **sizeD[]** Domain size in bits of the countable fields.

**Author**

vohuudtr

## 5.18.2 Member Function Documentation

### 5.18.2.1 static BigInteger **common.Parameters.getB** ( ) `[inline, static]`

Returns secret `b`.

**Returns**

secret `b`.

### 5.18.2.2 static int **common.Parameters.getMaxSizeD** ( ) `[inline, static]`

Returns the maximum domain size over all the countable fields.

**Returns**

maximum domain size.

#### 5.18.2.3 static BigInteger **common.Parameters.getN** ( ) `[inline, static]`

Returns upperbound of the number of records.

**Returns**

upperbound of the number of records.

#### 5.18.2.4 static int **common.Parameters.getNumFields** ( ) `[inline, static]`

Returns number of countable fields.

**Returns**

number of countable fields.

#### 5.18.2.5 static BigInteger **common.Parameters.getP** ( ) `[inline, static]`

Returns secret prime `p`.

**Returns**

secret prime `p`.

#### 5.18.2.6 static BigInteger **common.Parameters.getQ** ( ) `[inline, static]`

Returns prime `q`.

**Returns**

prime `q`.

#### 5.18.2.7 static int **common.Parameters.getS1** ( ) `[inline, static]`

Returns security parameter `s1`.

**Returns**

security parameter `s1`.

#### 5.18.2.8 static int **common.Parameters.getS2** ( ) `[inline, static]`

Returns security parameter `s2`.

**Returns**

security parameter `s2`.

#### 5.18.2.9 static int **common.Parameters.getSizeD** ( int *field* ) `[inline, static]`

Returns domain size of a field.

**Parameters**

| | |
|---|---|
| *field* | given field. |

**Returns**

domain size of the specified field.

#### 5.18.2.10 static int **common.Parameters.getTotalSizeD** ( ) `[inline, static]`

Returns the total domain size which is the product of domain size of all fields.

**Returns**

total domain size.

#### 5.18.2.11 static void **common.Parameters.main** ( String[] *args* ) `[inline, static]`

**Parameters**

| | |
|---|---|
| *args* | |

#### 5.18.2.12 static void **common.Parameters.readConfig** ( String *filename* ) `[inline, static]`

Reads parameters from a configuration file.

**Parameters**

| | |
|---|---|
| *filename* | name of the configuration file. |

#### 5.18.2.13 static void **common.Parameters.readConfig** ( InputStream *is* ) `[inline, static]`

Reads parameters from an input stream.

**Parameters**

| | |
|---|---|
| *is* | input stream. |

#### 5.18.2.14 static void **common.Parameters.setB** ( BigInteger *newb* ) `[inline, static]`

Sets the new value for `b`.

**Parameters**

| | |
|---|---|
| *newb* | new value for `b`. |

#### 5.18.2.15 static void **common.Parameters.setP** ( BigInteger *newp* ) `[inline, static]`

Sets new secret prime `p`.

**Parameters**

| | |
|---|---|
| *newp* | new secret prime `p`. |

#### 5.18.2.16 static void **common.Parameters.setQ** ( BigInteger *newq* ) `[inline, static]`

Sets new prime `q`.

**Parameters**

| | |
|---|---|
| *newq* | new prime `q`. |

#### 5.18.2.17 static void **common.Parameters.writeConfig** ( String *filename* ) `[inline, static]`

Writes parameters to a configuration file.

**Parameters**

| | |
|---|---|
| *filename* | name of the configuration file. |

#### 5.18.2.18 static void **common.Parameters.writeConfig** ( OutputStream *os* ) `[inline, static]`

Writes parameters to an output stream.

**Parameters**

| | |
|---|---|
| *os* | output stream. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Parameters.java

## 5.19 obsolete.PolyMatrix Class Reference

### Public Member Functions

- PolyMatrix (int size_D)

  *Constructor.*
- BigVector getColumn (int v)

  *Return a specified column in the polynomial matrix.*
- BigVector getRow (int v)

  *Return a specified row in the polynomial matrix.*
- Polynomial getPolynomial (int v)

  *Return the polynomial corresponding to v.*

### Static Public Member Functions

- static void **main** (String args[ ])

### 5.19.1 Constructor & Destructor Documentation

#### 5.19.1.1 obsolete.PolyMatrix.PolyMatrix ( int *size_D* ) `[inline]`

Constructor.

**Parameters**

| | |
|---|---|
| *size_D* | size of input domain D. |

### 5.19.2 Member Function Documentation

#### 5.19.2.1 BigVector obsolete.PolyMatrix.getColumn ( int *v* ) `[inline]`

Return a specified column in the polynomial matrix.

**Parameters**

| | |
|---|---|
| *v* | index of the requested column. |

**Returns**

vector representing the requested column.

#### 5.19.2.2 Polynomial obsolete.PolyMatrix.getPolynomial ( int *v* ) `[inline]`

Return the polynomial corresponding to v.

**Parameters**

| | |
|---|---|
| *v* | specified v. |

**Returns**

the polynomial corresponding to v.

#### 5.19.2.3 BigVector obsolete.PolyMatrix.getRow ( int *v* ) `[inline]`

Return a specified row in the polynomial matrix.

**Parameters**

| | |
|---|---|
| *v* | index of the requested row. |

**Returns**

vector representing the requested row.

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/obsolete/PolyMatrix.java

## 5.20 common.Polynomial Class Reference

Provides an implementation for a univariate polynomial `P(x)`.

### Public Member Functions

- Polynomial (int highestDegree)

  *Constructing the univariate polynomial with a given highest degree that the polynomial can have.*
- Polynomial (BigInteger[ ] coefficients)

  *Constructing the univariate polynomial with given coefficients in Java BigInteger type.*
- Polynomial (int[ ] coefficients)

  *Constructing the univariate polynomial with given coefficients in integer type.*
- int degree ()

  *Returns the polynomial degree.*
- BigInteger getCoefficient (int degree)

*Returns coefficient corresponding the given degree.*

- BigInteger[] getCoefficients ()

  *Returns all coefficients of the polynomial.*
- void setCoefficient (int degree, BigInteger coeff)

  *Sets coefficient at a specified degree with given value in Java BigInteger type.*
- void setCoefficient (int degree, int coeff)

  *Sets coefficient of a specified degree with given value in integer type.*
- BigInteger value (BigInteger x)

  *Evaluates the polynomial at a given point.*
- Polynomial add (Polynomial p)

  *Returns a new univariate polynomial as the sum of this polynomial and another polynomial.*
- Polynomial subtract (Polynomial p)

  *Returns a new univariate polynomial as the result of subtracting this polynomial by another polynomial.*
- Polynomial negate ()

  *Returns a new univariate polynomial by negating this polynomial.*
- Polynomial multiply (BigInteger k)

  *Returns a new univariate polynomial by multiplying this polynomial with a scalar value in Java BigInteger type.*
- Polynomial multiply (int k)

  *Returns a new univariate polynomial by multiplying this polynomial with a scalar value in integer type.*
- Polynomial multiply (Polynomial p)

  *Returns a new univariate polynomial by multiplying this polynomial with another polynomial.*
- Polynomial pow (int exponent)

  *Returns a new univariate polynomial as an exponentiation of this polynomial.*
- Polynomial mod (BigInteger m)

  *Returns a new univariate polynomial by mod-ing all coefficients of this polynomial by a given modulus.*
- String toString ()

  *Returns a string representing this polynomial.*

## Static Public Member Functions

- static Polynomial createLagrange (int[] x, int i)

  *Creates a univariate polynomial in Lagrange form.*
- static void main (String args[])

  *For testing purpose.*

### 5.20.1 Detailed Description

Provides an implementation for a univariate polynomial `P(x)`.

**Author**

vohuudtr

### 5.20.2 Constructor & Destructor Documentation

#### 5.20.2.1 common.Polynomial.Polynomial ( int *highestDegree* ) `[inline]`

Constructing the univariate polynomial with a given highest degree that the polynomial can have.

At initialization, all coefficients are set to 0, and the polynomial degree is 0.

**Parameters**

| | |
|---|---|
| *highest-Degree* | the highest degree. |

#### 5.20.2.2 common.Polynomial.Polynomial ( BigInteger[] *coefficients* ) `[inline]`

Constructing the univariate polynomial with given coefficients in Java BigInteger type.

The polynomial degree is determined based on the values of the given coefficients.

**Parameters**

| | |
|---|---|
| *coefficients* | given coefficients. |

#### 5.20.2.3 common.Polynomial.Polynomial ( int[] *coefficients* ) `[inline]`

Constructing the univariate polynomial with given coefficients in integer type.

The polynomial degree is determined based on the values of the given coefficients.

**Parameters**

| | |
|---|---|
| *coefficients* | given coefficients. |

### 5.20.3 Member Function Documentation

#### 5.20.3.1 Polynomial common.Polynomial.add ( Polynomial *p* ) [inline]

Returns a new univariate polynomial as the sum of this polynomial and another polynomial.

**Parameters**

| | |
|---|---|
| *p* | another polynomial. |

**Returns**

sum of the polynomials.

#### 5.20.3.2 static Polynomial common.Polynomial.createLagrange ( int[ ] *x*, int *i* ) [inline, static]

Creates a univariate polynomial in Lagrange form.

$l_i(x) = \prod_{m \neq i} \frac{x - x_m}{x_i - x_m}$

**Parameters**

| | |
|---|---|
| *x* | array of all points $x_m$. |
| *i* | index $i$ of $x_i$. |

#### 5.20.3.3 int common.Polynomial.degree ( ) [inline]

Returns the polynomial degree.

**Returns**

the polynomial degree.

#### 5.20.3.4 BigInteger common.Polynomial.getCoefficient ( int *degree* ) [inline]

Returns coefficient corresponding the given degree.

**Parameters**

| | |
|---|---|
| *degree* | degree of the queried coefficient. |

**Returns**

the queried coefficient. If the queried degree is greater than the polynomial degree, 0 is returned.

#### 5.20.3.5 BigInteger [ ] common.Polynomial.getCoefficients ( ) `[inline]`

Returns all coefficients of the polynomial.

**Returns**

array of all coefficients.

#### 5.20.3.6 static void common.Polynomial.main ( String *args[]* ) `[inline, static]`

For testing purpose.

**Parameters**

| | |
|---|---|
| *args* | |

#### 5.20.3.7 Polynomial common.Polynomial.mod ( BigInteger *m* ) `[inline]`

Returns a new univariate polynomial by mod-ing all coefficients of this polynomial by a given modulus.

**Parameters**

| | |
|---|---|
| *m* | the modulus. |

**Returns**

result of the mod operation.

#### 5.20.3.8 Polynomial common.Polynomial.multiply ( BigInteger *k* ) `[inline]`

Returns a new univariate polynomial by multiplying this polynomial with a scalar value in Java BigInteger type.

**Parameters**

| | |
|---|---|
| *k* | scalar value. |

**Returns**

result of the multiplication.

#### 5.20.3.9 Polynomial common.Polynomial.multiply ( int *k* ) `[inline]`

Returns a new univariate polynomial by multiplying this polynomial with a scalar value in integer type.

**Parameters**

| | |
|---|---|
| *k* | scalar value. |

**Returns**

result of the multiplication.

#### 5.20.3.10 Polynomial common.Polynomial.multiply ( Polynomial *p* ) `[inline]`

Returns a new univariate polynomial by multiplying this polynomial with another polynomial.

The multiplication is done among coefficients of two polynomials.

**Parameters**

| | |
|---|---|
| *p* | another polynomial. |

**Returns**

result of the multiplication.

#### 5.20.3.11 Polynomial common.Polynomial.negate ( ) `[inline]`

Returns a new univariate polynomial by negating this polynomial.

**Returns**

the negation of this polynomial.

#### 5.20.3.12 Polynomial common.Polynomial.pow ( int *exponent* ) `[inline]`

Returns a new univariate polynomial as an exponentiation of this polynomial.

**Parameters**

| | |
|---|---|
| *exponent* | desired exponent. |

**Returns**

result of the exponentiation.

#### 5.20.3.13 void **common.Polynomial.setCoefficient** ( int *degree*, BigInteger *coeff* ) `[inline]`

Sets coefficient at a specified degree with given value in Java BigInteger type.

**Parameters**

| | |
|---|---|
| *degree* | specified degree. |
| *coeff* | new value for the coefficient. |

#### 5.20.3.14 void **common.Polynomial.setCoefficient** ( int *degree*, int *coeff* ) `[inline]`

Sets coefficient of a specified degree with given value in integer type.

**Parameters**

| | |
|---|---|
| *degree* | specified degree. |
| *coeff* | new value for the coefficient. |

#### 5.20.3.15 **Polynomial common.Polynomial.subtract** ( **Polynomial** *p* ) `[inline]`

Returns a new univariate polynomial as the result of subtracting this polynomial by another polynomial.

**Parameters**

| | |
|---|---|
| *p* | another polynomial. |

**Returns**

subtraction of this polynomial by another polynomial.

#### 5.20.3.16 BigInteger common.Polynomial.value ( BigInteger *x* ) `[inline]`

Evaluates the polynomial at a given point.

**Parameters**

| | |
|---|---|
| *x* | given value `x`. |

**Returns**

`P(x).`

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Polynomial.java

## 5.21 common.Producer Class Reference

Implementation of a local data generator.

### Static Public Member Functions

- static Record generateRecord ()

  *Generate a random record.*
- static Statistics getStats ()

  *Returns the statistics of generated records.*
- static void main (String[ ] args)

  *Usage:*

### Static Public Attributes

- static OutputStream **paramos** = null
- static OutputStream **statos** = null

### Static Package Functions

- static void **generate** (long quantity) throws IOException
- static void **generate** (long quantity, boolean split) throws IOException
- static void **generate** (long quantity, OutputStream os) throws IOException

### 5.21.1 Detailed Description

Implementation of a local data generator.

This class is used for experiments.

A record is formatted in order of columns as follows:

- Encrypted countable field 1.
- Plain-text countable field 1.
- Encrypted countable field 2.

- Plain-text countable field 2. ...
- Encrypted countable field m.
- Plaini-text countable field m.
- Remaining random data

Notes:

- Size of each record is specified in Record.
- Number of countable fields, m, is specified in the parameters file (see Parameters).
- Size of each encrypted countable field depends on size of prime p (see Parameters). Encrypted countable field's format is defined in BigIntegerWritable class.

**See also**

main(String[]) for detailed usage.

**Author**

vohuudtr

## 5.21.2 Member Function Documentation

### 5.21.2.1 static Record common.Producer.generateRecord ( ) `[inline, static]`

Generate a random record.

While generating records, statistics about the values are also recorded for evaluation.

**Returns**

the generated record.

### 5.21.2.2 static Statistics common.Producer.getStats ( ) `[inline, static]`

Returns the statistics of generated records.

**Returns**

statistics of generated records.

#### 5.21.2.3 static void common.Producer.main ( String[] *args* ) `[inline, static]`

Usage:

```
java common.Producer <quantity> <paramfile> <statfile>
[datafile]
```

**quantity** Number of records to generate.

**paramfile** Path to the parameters file (see Parameters).

**statfile** Path to the output statistics file.

**datafile** Path to the output data file. The output is automatically split into multiple files (suffixed with numbers) of size not greater than 1GB. If `datafile` is not specified, the standard output is used.

All the paths can be either absolute or relative paths.

**Parameters**

| | |
|---|---|
| *args* | |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Producer.java

## 5.22 mapred.Producer Class Reference

This is a MapReduce job used for generating a large data of set and storing in the HDFS.

Inherits Configured, and Tool.

### Classes

- class **AmazonProducerInputFormat**
- class **AmazonProducerOutputFormat**
- class **AmazonProducerRecordReader**
- class **AmazonProducerRecordWriter**
- class **AmazonProducerSplit**
- class **Map**

### Public Member Functions

- int run (String[] args) throws Exception

  *main function*

### Static Public Member Functions

- static void **main** (String[ ] args) throws Exception

### 5.22.1 Detailed Description

This is a MapReduce job used for generating a large data of set and storing in the HDFS.

The generated data in the HDFS will be read later by a counting job. This class was created and used for experiments.

Usage:

```
hadoop jar <JARFILE> mapred.Producer -Dquantity=<quantity>
-Dparamfile=<paramfile> <genoutput>
```

**JARFILE** The JAR file containing the application.

**quantity** Number of records to generate.

**paramfile** Name of parameters file. Path to the parameters file must be a relative path to this class in the JAR file. See Parameters for details on parameters file.

**genoutput** A HDFS path to the directory containing the generated data.

For details on what data is generated and how records are formatted, see common.Producer.

**Author**

vohuudtr

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/Producer.java

## 5.23 common.Record Class Reference

Definition of a record used by the application.

Inherits Writable.

### Public Member Functions

- Record ()

  *Initializes the record with zero fields.*
- Record (int numFields)

  *Initializes the record with a given number of fields.*

- void readFields (DataInput in) throws IOException

  *Re-initializes the record by reading from the input stream.*
- void write (DataOutput out) throws IOException

  *Writes the record to the given output stream using the format specified in readFields(-DataInput).*
- int getSize ()

  *Returns total size in bytes of all fields in the record, not including the rest of random bytes.*
- BigIntegerWritable getField (int i)

  *Return values of the specified field.*
- void setField (int i, BigIntegerWritable value)

  *Set value of the specified field.*
- int getNumFields ()

  *Returns the number of fields in the record (not considering random bytes of the rest).*

### Static Public Attributes

- static int blocksize = 1000000

  *Specifies a fixed size in bytes for each record.*

## 5.23.1 Detailed Description

Definition of a record used by the application.

Each record has a fixed size that can be changed via blocksize.

Each record also has a fixed number of fields.

*Note:* for convenient experiments, a generated record contains twice the number of countable fields defined in the parameter file, i.e. if the number of countable fields is m, the actual number of fields defined in each record is 2m, where an even-indexed 2k-th field contains the encrypted value of the k-th field, while an odd-indexed (2k+1)-th field contains the plain-text value of the k-th field.

**Author**

vohuudtr

## 5.23.2 Constructor & Destructor Documentation

### 5.23.2.1 common.Record.Record ( int *numFields* ) `[inline]`

Initializes the record with a given number of fields.

**Parameters**

| | |
|---|---|
| *numFields* | number of fields. |

### 5.23.3 Member Function Documentation

#### 5.23.3.1 BigIntegerWritable common.Record.getField ( int *i* ) `[inline]`

Return values of the specified field.

**Parameters**

| | |
|---|---|
| *i* | index of the field. |

**Returns**

value of the field.

#### 5.23.3.2 int common.Record.getNumFields ( ) `[inline]`

Returns the number of fields in the record (not considering random bytes of the rest).

**Returns**

number of fields.

#### 5.23.3.3 int common.Record.getSize ( ) `[inline]`

Returns total size in bytes of all fields in the record, not including the rest of random bytes.

**Returns**

total size.

#### 5.23.3.4 void common.Record.readFields ( DataInput *in* ) throws IOException `[inline]`

Re-initializes the record by reading from the input stream.

Format of record in the input is as follows:

- First 4 bytes represent an integer, which specifies the number of fields (2m).
- Field 1 (i.e. encrypted field 1).
- Field 2 (i.e. plain-text field 1).
- Field 3 (i.e. encrypted field 2).
- Field 4 (i.e. plain-text field 2).

- ...
- Field 2m (i.e. encrypted field m).
- Field 2m+1 (i.e. plain-text field m).
- Remaining random bytes for other fields that are not considered.

All fields are in BigIntegerWritable format.

#### 5.23.3.5 void **common.Record.setField** ( int *i*, **BigIntegerWritable** *value* ) [inline]

Set value of the specified field.

**Parameters**

| | |
|---|---|
| *i* | index of the field. |
| *value* | value of the field. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Record.java

## 5.24 mapred.RecordInputFormat Class Reference

Provides customized input format for MapReduce counting jobs.

Inherits FileInputFormat< LongWritable, Record >.

### Public Member Functions

- RecordReader< LongWritable, Record > **getRecordReader** (InputSplit split, JobConf conf, Reporter reporter) throws IOException

### 5.24.1 Detailed Description

Provides customized input format for MapReduce counting jobs.

**Author**

vohuudtr

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/mapred/RecordInputFormat.java

## 5.25 common.Statistics Class Reference

Provides a tool for collecting statistics about the values of fields in the data set.

### Public Member Functions

- Statistics ()

  *Constructs the object without initialization.*
- void initialize (int size)

  *Initializes the object with total number of distinct values in the data set.*
- void add (Record record)

  *Add and parse the record, and update the statistics.*
- void add (MultiFieldKey key, int quantity)

  *Add explicitly a value with given quantity.*
- void print (OutputStream out)

  *Prints statistics to given output stream.*
- void print ()

  *Prints statistics to standard output stream.*
- void print (String filename)

  *Print statistics to a given file.*

### 5.25.1 Detailed Description

Provides a tool for collecting statistics about the values of fields in the data set.

**Author**

vohuudtr

### 5.25.2 Member Function Documentation

#### 5.25.2.1 void **common.Statistics.add ( Record *record* )** `[inline]`

Add and parse the record, and update the statistics.

**Parameters**

| | |
|---|---|
| *record* | record to add. |

#### 5.25.2.2 void **common.Statistics.add ( MultiFieldKey *key*,** int ***quantity* )** `[inline]`

Add explicitly a value with given quantity.

**Parameters**

| | |
|---|---|
| *key* | |
| *quantity* | |

#### 5.25.2.3 void **common.Statistics.initialize** ( int *size* ) `[inline]`

Initializes the object with total number of distinct values in the data set.

**Parameters**

| | |
|---|---|
| *size* | number of distinct values in the data set. |

#### 5.25.2.4 void **common.Statistics.print** ( OutputStream *out* ) `[inline]`

Prints statistics to given output stream.

**Parameters**

| | |
|---|---|
| *out* | output stream. |

#### 5.25.2.5 void **common.Statistics.print** ( String *filename* ) `[inline]`

Print statistics to a given file.

**Parameters**

| | |
|---|---|
| *filename* | output filename. |

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

- /home/vohuudtr/workspace/cloudprivacy/code/src/common/Statistics.java