Package 'boolfun'

December 13, 2009

Title Cryptographic Boolean Functions

Version 0.2.1

Date 2009

Author F.Lafitte

Description This package can be used to assess cryptographic properties of Boolean functions (such as nonlinearity, algebraic immunity, resiliency, ...).

Maintainer Frederic Lafitte <frederic.lafitte@rma.ac.be>

Depends R (>= 2.3.0), R.oo

License GPL (>= 3)

Keywords package, math, logic

R topics documented:

`boolfun-package` *Cryptographic Boolean Functions*

Description

This package can be used to assess cryptographic properties of Boolean functions (such as nonlinearity, algebraic immunity, resiliency, ...).

Details

Package:	boolfun
Version:	0.2.1
Date:	2009
Depends:	R (>= 2.3.0), R.oo
License:	GPL (>= 3)
Built:	R 2.9.1; i686-pc-linux-gnu; 2009-12-10 13:57:37 UTC; unix

Index:

Further information is available in the following vignettes:

`boolfun`	Cryptographic Properties of Boolean functions (source, pdf)

See the examples below for an overview of how to use the package.

Author(s)

F.Lafitte

Maintainer: Frederic Lafitte <frederic.lafitte@rma.ac.be>

See Also

`R.oo`

Examples

```
# Functions are defined by their truth tables (string or integer vector).
f <- BooleanFunction( "00100111" )
g <- BooleanFunction(c(0,1,1,0,1,0,0,1))
h <- BooleanFunction( c(tt(f), tt(g)) ) # concatenation

# You can print information on the function as follows.
print ( h )     # Prints "Boolean function with 4 variables".
print ( tt(h) ) # Prints the truth table.

# Note that the methods can be called 'object$method()' or 'method(object)':
print(paste(  "f has (deg,ai,nl,res) = (", f$deg(),f$ai(),f$nl(),f$res(),")"  ))
print(paste(  "h has (deg,ai,nl,res) = (", deg(h), ai(h), nl(h), res(h), ")"  ))

# Random Boolean functions
randomBFs <- c()
data <- c( "degree", "algebraic immunity", "nonlinearity", "resiliency" )
for( i in 1:500 ) {
    randomTT <- round(runif(2^5, 0,1))
    randomBF <- BooleanFunction(randomTT)
```

```
    data <- rbind( data, c(
            deg(randomBF), ai(randomBF),
            nl(randomBF), res(randomBF)))
}
print(data)
```

`ai.BooleanFunction` *Algebraic Immunity*

Description

Returns the algebraic immunity - see Details.

Usage

```
## S3 method for class 'BooleanFunction':
ai(this, ...)
```

Arguments

`this`	-
`...`	Not used

Details

The algebraic immunity is obtained by gaussian elimination using C++ code. For more information, type `vignette(boolfun)`.

Value

This method returns the smallest degree (integer) of a non-zero annihilator of the function (f) or its complement (1+f).

See Also

`BooleanFunction`

`BooleanFunction` *Cryptographic Boolean Functions*

Description

The class `BooleanFunction` implements functionality to assess cryptographic properties of Boolean functions such as nonlinearity, algebraic immunity, resiliency, correlation immunity, ... For a full list, type `library(help=boolfun)`. Future versions will implement more functionality.

Usage

```
f <- BooleanFunction( initializer )
```

Arguments

`initializer`	a vector containing 2^n integers in $\{0, 1\}$ or a string holding 2^n characters in $\{'0', '1'\}$.

Details

The representations are computed in $\mathcal{O}(n2^n)$ using C++ code. They are computed only once, the first time they are needed/called by the user and stored in private fields. The same applies to some properties, namely algebraic immunity, algebraic degree and correlation immunity. Efforts have been made to optimize execution time rather than memory usage. For more details, see the package vignette (using the R command `vignette(boolfun)`) or see the examples below.

Value

The returned value `f` is an S3 object which is defined using the R.oo package. Methods of the returned value, say `deg()`, can be accessed in two ways using `f$deg()` or `deg(f)`.

Private Fields

Heavy computations are carried only once, the first time they are needed/called. The results are stored/cached in private fields.
Private fields:

`.ANF`	the algebraic normal form (vector)
`.WH`	the walsh spectrum (vector)
`.TT`	the truth table (vector)
`.deg`	the algebraic degree (integer)
`.ai`	the algebraic immunity (integer)
`.ci`	the correlation immunity (integer)

Note: R does not block access to private fields. However, using them is not recommended, use the accessors (public methods) instead.

Public Methods

Three (unique) representations are implemented:

`tt()`	returns the truth table (vector of integers)
`wh()`	returns the walsh spectrum (vector of integers)
`anf()`	returns the vector of coefficients of each of the 2^n monomials - see `mobiusTransform`

Some general properties of the boolean function:

`n()`	returns the number of input variables (i.e. $f : \{0,1\}^n \rightarrow \{0,1\}$)
`deg()`	returns the algebraic degree of the boolean function

Some properties relevant for cryptographic applications:

`ai()`	returns the algebraic immunity
`nl()`	returns the nonlinearity
`ci()`	returns the correlation immunity
`res()`	returns the resiliency

Some conditionals:

`isBal()`	returns true if the function has as many 0s as 1s in its truth table
`isCi(t)`	returns true if the function has correlation immunity `t`
`isRes(t)`	returns true if the function has resiliency `t`

Overridden Methods

`BooleanFunction` inherits from `Object` defined in the R.oo package. The following methods are overriden:

`equals()`	compares truth tables and returns true if they are the same
`print()`	displays the number of variables
`hashCode()`	returns `Object`'s `hashCode` of the truth table

Author(s)

F.Lafitte

See Also

`mobiusTransform`, `walshTransform`, `R.oo:Object`

Examples

```
truthTable <- c(0,1,1,0,1,0,0,1)
f <- BooleanFunction(truthTable)
g <- BooleanFunction("00100111")
h <- BooleanFunction( c( tt(f), tt(g) ) ) # concatenation
print( h )
#
print( paste("f has (deg,ai,nl,res) = (", deg(f), ai(f), nl(f), res(f), ")" ))
print( paste("g has (deg,ai,nl,res) = (", g$deg(), g$ai(), g$nl(), g$res(), ")" ))
print( isBal(h) )
```

`mobiusTransform` *Mobius Inversion*

Description

In this package, the Mobius inversion is used to compute the coefficient of each monomial in the algebraic normal form of the input Boolean function. That is, `mobiusTransforms` returns a vector of length 2^n where each entry equals one if the corresponding monomial appears in the algebraic normal form (zero otherwise) - see Details.

Usage

```
mobiusTransform( truthTable )
```

Arguments

`truthTable`	a vector of integers containing 2^n binary entries - see `BooleanFunction`.

Details

The value is computed in $\mathcal{O}(n2^n)$. For more information, type `vignette(boolfun)`.

Value

`mobiusTransform` returns a vector of integers in $\{0, 1\}$ of same length as the input vector (i.e. 2^n). The ith entry is 1 if the monomial i appears in the algebraic normal form of the input function.

References

Graham, Knuth, Patashnik. Concrete Mathematics. second edition. pp.136.

Ann Braeken. Cryptographic Properties of Boolean Functions and S-Boxes. phd thesis - 2006.

The vignette of this package.

See Also

`BooleanFunction`

Examples

```
tt <- c(0,1,1,0,1,0,0,1)
anf <- mobiusTransform(tt)
```

`utils` *Some Auxiliary Functions*

Description

Some functions used by the `BooleanFunction` object that might be useful for other purposes.

Usage

```
toBin(a,n)
modulo(a,n)
weight(x)
```

Arguments

`a`	an integer.
`n`	an integer.
`x`	an integer or a vector of integers. If `x` is a vector the function will be applied to each of its components.

Value

`toBin` returns a binary representation of `x` as a vector of `n` integers. Note that the binary representation is reversed, that is, `toBin(8,4)` returns `(0,0,0,1)` instead of `(1,0,0,0)`.
`modulo` returns `a` mod `n`.
`weight` returns the hamming weight of the binary representation of `x`. If `x` is a vector, the hamming weight of each of its components is returned in a vector.

Author(s)

F.Lafitte

See Also

`BooleanFunction`

Examples

```
powers <- c( 2, 4, 8, 16, 32,  64 )
if( any( weight(powers) != 1 ) )
    stop("This message should not print")
for( i in 0:(2^10 -1) )
    if( sum(toBin(i,10)) != weight(i) )
        stop("This message should not print")
```

`walshTransform` *Fast Walsh Hadamard Transform*

Description

`walshTransform` returns the Walsh-Hadamard transform of the input truth table.

Usage

```
walshTransform( truthTable )
```

Arguments

`truthTable`	a vector of integers containing 2^n binary entries - see `BooleanFunction`.

Details

The value is computed in $\mathcal{O}(n2^n)$ using the Fast Walsh Transform (FWT). For more information, type `vignette(boolfun)`.

Value

`walshTransform` returns a vector of integers having the same length as the input vector. The i^{th} entry can be seen as a "similarity" or "association" with the linear function determined by the integer i (i.e. there are 2^n such functions).

References

James L. Massey. The Discrete Fourier Transform in Coding and Cryptography. IEEE Inform. Theory Workshop, ITW 1998, pages 9–11.

Ann Braeken. Cryptographic Properties of Boolean Functions and S-Boxes. phd thesis - 2006.

The vignette of this package.

See Also

`BooleanFunction`

Examples

```
tt <- c(0,1,1,0,1,0,0,1)
wh <- walshTransform(tt)
```

Index