PySNARK Documentation

Release

Meilof Veeningen

Jan 04, 2018

CONTENTS:

1 pysnark package **1**
- 1.1 Subpackages . . . 1
- 1.2 Submodules . . . 3
- 1.3 Module contents . . . 6

2 Indices and tables **7**

Python Module Index **9**

Index **11**

CHAPTER

ONE

PYSNARK PACKAGE

1.1 Subpackages

1.1.1 pysnark.lib package

Submodules

pysnark.lib.array module

class pysnark.lib.array.**Array**(*vals*)

assert_equals(*other*)

joined()

pysnark.lib.base module

pysnark.lib.base.**if_then_else**(*cond*, *trueval*, *falseval*)
Returns one of two values depending on choice bit :param cond: Choice bit (function does not ensure that this is a bit) :param trueval: Value if choice bit is 1 :param falseval: Value if choice bit is 0 :return: Value given by choice bit

pysnark.lib.base.**input_bit_array**(*bits*, *nm=None*)
Imports bitstring as a single input of the program. This checks that all provided values are actually bits.

Parameters

- **bits** – String consisting of (at most 253) 0s and 1s (starting with most significant)
- **nm** – Name of input variable (None for automatic)

Returns Array of bits

pysnark.lib.base.**lin_comb**(*cofs*, *vals*)
Returns linear combination of given values with given coefficients :param cofs: Array of variable coefficients :param vals: Array of variable values :return: Variable representing the linear combination

pysnark.lib.base.**lin_comb_pub**(*cofs*, *vals*)
Returns linear combination of given values with given coefficients. This can be executed more efficiently than computing the sum by hand but introduces an additional equation to the program.

Parameters

- **cofs** – Array of integer coefficients

- **vals** – Array of variable values

Returns Variable representing the linear combination

pysnark.lib.base.**output_bit_array**(*bits*, *nm=None*)

Exports bitstring as a single output of the QAP. This does not check that all provided values are actually bits; use *pysnark.runtime.Var.assert_bit()* for that.

Parameters

- **bits** – Array of Vars representing bits
- **nm** – Name of output variable (None for automatic)

Returns Bitstring representing the value

pysnark.lib.fixedpoint module

class pysnark.lib.fixedpoint.**VarFxp**(*val*, *sig=None*)

Bases: *pysnark.runtime.Var*

Variable representing a fixed-point number. Number x is represented as integer $x * 2^r$, where r is the resolution *VarFxp.res*

floatval()

Returns floating-point value represented by this variable.

Returns value

classmethod fromvar(*var*)

Convers a non-fixed-point variable to fixed point

Parameters **var** – A non-fixed-point variable

Returns A new fixed-point variable representing the same value

classmethod fromvar_noconv(*var*)

maxden = 40

Maximal length of denominiators for __div__(), including resolution

res = 20

Resulution for fixed-point numbers

val(*nm=None*)

pysnark.lib.ggh module

This module provides efficient hashes using the "Ajtai-GGH" hash function due to Ajtai and Goldreich/Goldwasser/Halevi (Goldreich, Goldwasser, Halevi, "Collision-Free Hashing from Lattice Problems").

This implementation uses parameters N=64, Q=524288, M=7296 and translates a 7296-bit input into a 1216-bit output (i.e., it has a compression ratio of 6). These parameters are as suggested by Chris Peikert and used in Pepper, see here.

The coefficients of the hash function have been generated using a PRF.

pysnark.lib.ggh.**ggh_hash**(*plain*)

Computes a GGH hash of the given input bits. This function does not ensure that the inputs are actually bits, but it guarantees that the outputs are bits

Parameters **plain** – Plaintext: array of 7296 bits

Returns Hash: array of 1216 bits

pysnark.lib.ggh.**ggh_hash_packed**(*plain_packed*)

pysnark.lib.ggh_plain module

pysnark.lib.ggh_plain.**fromint**(*val*, *len*)

pysnark.lib.ggh_plain.**ggh_hash**(*plain*)

pysnark.lib.ggh_plain.**packin**(*vals*)

pysnark.lib.ggh_plain.**packout**(*vals*)

pysnark.lib.ggh_plain.**toint**(*vals*)

pysnark.lib.ggh_plain.**unpackin**(*vals*)

pysnark.lib.ggh_plain.**unpackout**(*vals*)

Module contents

1.2 Submodules

1.2.1 pysnark.atexitmaybe module

class pysnark.atexitmaybe.**ExitOverrider**

Bases: object

excepthook(*tp*, *ex*, **args*)

exit(*exitcode=0*)

pysnark.atexitmaybe.**maybe**(*fn*)

1.2.2 pysnark.contract module

class pysnark.contract.**QapVk**(*fn*)

pysnark.contract.**contract**()

pysnark.contract.**readg1**(*fl*)

pysnark.contract.**readg2**(*fl*)

pysnark.contract.**strg1**(*val*)

pysnark.contract.**strg1p**(*ix*)

pysnark.contract.**strg2**(*val*)

pysnark.contract.**strg2p**(*ix*)

pysnark.contract.**tog1**(*tok*)

pysnark.contract.**tog2**(*tok*)

1.2.3 pysnark.import module

1.2.4 pysnark.options module

pysnark.options.**do_proof**()

pysnark.options.**do_rebuild**()

pysnark.options.**get_block_comm**(*bname*)

pysnark.options.**get_block_file**(*bname*)

pysnark.options.**get_cache_file**(*sz*)

pysnark.options.**get_contract_dir**()

pysnark.options.**get_conttest_dir**()

pysnark.options.**get_ek_file**(*fn*)

pysnark.options.**get_eqs_file**()

pysnark.options.**get_eqs_file_fn**(*fn*)

pysnark.options.**get_from_environ**(*nm*, *default*)

pysnark.options.**get_io_file**()

pysnark.options.**get_mkey_file**()

pysnark.options.**get_mpkey_file**()

pysnark.options.**get_mskey_file**()

pysnark.options.**get_proof_file**()

pysnark.options.**get_qaptool_exe**(*tool*)

pysnark.options.**get_schedule_file**()

pysnark.options.**get_vk_file**(*fn*)

pysnark.options.**get_wire_file**()

pysnark.options.**vc_p = 21888242871839275222246405745257275088548364400416034343698204186575**

The modulus used in the verifiable computation. All computations are performed using modular arithmetic with this modulus.

1.2.5 pysnark.prove module

pysnark.prove.**prove**()

1.2.6 pysnark.qapsplit module

pysnark.qapsplit.**contextualize**(*lst*)

pysnark.qapsplit.**getqap**(*nm*)

pysnark.qapsplit.**qapsplit**()

Returns (maximum qap size, maximum input block size) encountered

1.2.7 pysnark.qaptools module

1.2.8 pysnark.runtime module

class pysnark.runtime.**Var**(**args*, ***kwargs*)

A variable of the verifiable computation

assert_bit()

Assert that this variable contains a bit, i.e., 0 or 1 :return: None

assert_equals(*other*)

assert_nonzero()

assert_positive(*bl*)

Assert that the present VcShare represents a positive value, that is, a value in [0,2^bl] with bl the given bit length.

assert_smaller(*val*)

assert_zero()

Assert that the present VcShare represents the value zero.

bit_decompose(*bl*)

Assert that the present VcShare represents a positive value, that is, a value in [0,2^bl] with bl the given bit length.

classmethod constant(*val*)

Return a VcShare representing the given constant value.

classmethod constname(**args*, ***kwargs*)

divmod(*divisor*, *maxquotbl*)

Divide by public value and return quotient and remainder :param divisor: Divisor (integer) :param maxquotbl: Maximal bitlength of the resulting quotient :return: Quotient and remainder

ensure_single()

Return a VcShare with the same value that is guaranteed to refer to one witness, by making a new VcShare and adding the required equation if necessary.

equals(*other*)

isnonzero()

Returns VcShare equal to 1 if self is not zero, and 0 if self is zero.

iszero()

classmethod random()

Return a VcShare representing a random value.

strsig()

Return string representation of linear combination represented by this VcShare.

classmethod tovar(*val*, *nm=None*)

val(**args*, ***kwargs*)

classmethod vals(*vars*, *nm*)

classmethod vars(*vals*, *nm*, *dim=1*)

classmethod zero()

Return a VcShare representing the value zero.

pysnark.runtime.**continuefn**(**args*, ***kwargs*)

pysnark.runtime.**enterfn**(*fname*, *call=None*)
Start a new call of the given function type :param fname: Function name. All instances of the same function should execute the exact same sequence of instructions :param call: Call name. Should be globally unique (autogenerated if not given) :return: Call name

pysnark.runtime.**exportarray**(**args*, ***kwargs*)

pysnark.runtime.**for_each_in**(*cls*, *f*, *struct*)
Recursively traversing all lists and tuples in struct, apply f to each element that is an instance of cls. Returns structure with f applied.

pysnark.runtime.**importarray**(**args*, ***kwargs*)

pysnark.runtime.**init**()

pysnark.runtime.**inited**(*fn*)

pysnark.runtime.**printwire**(**args*, ***kwargs*)

pysnark.runtime.**printwireout**(**args*, ***kwargs*)

pysnark.runtime.**subqap**(*nm*)

pysnark.runtime.**vc_assert_mult**(**args*, ***kwargs*)

pysnark.runtime.**vc_declare_block**(**args*, ***kwargs*)

pysnark.runtime.**vc_glue**(*ctx1*, *ctx2*, *vals*)

1.2.9 pysnark.testqap module

This tool tests whether the wire file in the current location (as given by *pysnark.options.get_wire_file()*) satisfies all equations of the current Quadratic Arithmetic Program (as given by *pysnark.options.get_eqs_file()*)

Run with

python -m pysnark.testqap

1.3 Module contents

The PySNARK package contains the main functionality of PySNARK.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

pysnark, 6
pysnark.atexitmaybe, 3
pysnark.contract, 3
pysnark.import, 4
pysnark.lib, 3
pysnark.lib.array, 1
pysnark.lib.base, 1
pysnark.lib.fixedpoint, 2
pysnark.lib.ggh, 2
pysnark.lib.ggh_plain, 3
pysnark.options, 4
pysnark.prove, 4
pysnark.qapsplit, 4
pysnark.qaptools, 5
pysnark.runtime, 5
pysnark.testqap, 6

INDEX

A

Array (class in pysnark.lib.array), 1
assert_bit() (pysnark.runtime.Var method), 5
assert_equals() (pysnark.lib.array.Array method), 1
assert_equals() (pysnark.runtime.Var method), 5
assert_nonzero() (pysnark.runtime.Var method), 5
assert_positive() (pysnark.runtime.Var method), 5
assert_smaller() (pysnark.runtime.Var method), 5
assert_zero() (pysnark.runtime.Var method), 5

B

bit_decompose() (pysnark.runtime.Var method), 5

C

constant() (pysnark.runtime.Var class method), 5
constname() (pysnark.runtime.Var class method), 5
contextualize() (in module pysnark.qapsplit), 4
continuefn() (in module pysnark.runtime), 5
contract() (in module pysnark.contract), 3

D

divmod() (pysnark.runtime.Var method), 5
do_proof() (in module pysnark.options), 4
do_rebuild() (in module pysnark.options), 4

E

ensure_single() (pysnark.runtime.Var method), 5
enterfn() (in module pysnark.runtime), 6
equals() (pysnark.runtime.Var method), 5
excepthook() (pysnark.atexitmaybe.ExitOverrider method), 3
exit() (pysnark.atexitmaybe.ExitOverrider method), 3
ExitOverrider (class in pysnark.atexitmaybe), 3
exportarray() (in module pysnark.runtime), 6

F

floatval() (pysnark.lib.fixedpoint.VarFxp method), 2
for_each_in() (in module pysnark.runtime), 6
fromint() (in module pysnark.lib.ggh_plain), 3
fromvar() (pysnark.lib.fixedpoint.VarFxp class method), 2
fromvar_noconv() (pysnark.lib.fixedpoint.VarFxp class method), 2

G

get_block_comm() (in module pysnark.options), 4
get_block_file() (in module pysnark.options), 4
get_cache_file() (in module pysnark.options), 4
get_contract_dir() (in module pysnark.options), 4
get_conttest_dir() (in module pysnark.options), 4
get_ek_file() (in module pysnark.options), 4
get_eqs_file() (in module pysnark.options), 4
get_eqs_file_fn() (in module pysnark.options), 4
get_from_environ() (in module pysnark.options), 4
get_io_file() (in module pysnark.options), 4
get_mkey_file() (in module pysnark.options), 4
get_mpkey_file() (in module pysnark.options), 4
get_mskey_file() (in module pysnark.options), 4
get_proof_file() (in module pysnark.options), 4
get_qaptool_exe() (in module pysnark.options), 4
get_schedule_file() (in module pysnark.options), 4
get_vk_file() (in module pysnark.options), 4
get_wire_file() (in module pysnark.options), 4
getqap() (in module pysnark.qapsplit), 4
ggh_hash() (in module pysnark.lib.ggh), 2
ggh_hash() (in module pysnark.lib.ggh_plain), 3
ggh_hash_packed() (in module pysnark.lib.ggh), 3

I

if_then_else() (in module pysnark.lib.base), 1
importarray() (in module pysnark.runtime), 6
init() (in module pysnark.runtime), 6
inited() (in module pysnark.runtime), 6
input_bit_array() (in module pysnark.lib.base), 1
isnonzero() (pysnark.runtime.Var method), 5
iszero() (pysnark.runtime.Var method), 5

J

joined() (pysnark.lib.array.Array method), 1

L

lin_comb() (in module pysnark.lib.base), 1
lin_comb_pub() (in module pysnark.lib.base), 1

M

maxden (pysnark.lib.fixedpoint.VarFxp attribute), 2

maybe() (in module pysnark.atexitmaybe), 3

O

output_bit_array() (in module pysnark.lib.base), 2

P

packin() (in module pysnark.lib.ggh_plain), 3
packout() (in module pysnark.lib.ggh_plain), 3
printwire() (in module pysnark.runtime), 6
printwireout() (in module pysnark.runtime), 6
prove() (in module pysnark.prove), 4
pysnark (module), 6
pysnark.atexitmaybe (module), 3
pysnark.contract (module), 3
pysnark.import (module), 4
pysnark.lib (module), 3
pysnark.lib.array (module), 1
pysnark.lib.base (module), 1
pysnark.lib.fixedpoint (module), 2
pysnark.lib.ggh (module), 2
pysnark.lib.ggh_plain (module), 3
pysnark.options (module), 4
pysnark.prove (module), 4
pysnark.qapsplit (module), 4
pysnark.qaptools (module), 5
pysnark.runtime (module), 5
pysnark.testqap (module), 6

Q

qapsplit() (in module pysnark.qapsplit), 4
QapVk (class in pysnark.contract), 3

R

random() (pysnark.runtime.Var class method), 5
readg1() (in module pysnark.contract), 3
readg2() (in module pysnark.contract), 3
res (pysnark.lib.fixedpoint.VarFxp attribute), 2

S

strg1() (in module pysnark.contract), 3
strg1p() (in module pysnark.contract), 3
strg2() (in module pysnark.contract), 3
strg2p() (in module pysnark.contract), 3
strsig() (pysnark.runtime.Var method), 5
subqap() (in module pysnark.runtime), 6

T

tog1() (in module pysnark.contract), 3
tog2() (in module pysnark.contract), 3
toint() (in module pysnark.lib.ggh_plain), 3
tovar() (pysnark.runtime.Var class method), 5

U

unpackin() (in module pysnark.lib.ggh_plain), 3
unpackout() (in module pysnark.lib.ggh_plain), 3

V

val() (pysnark.lib.fixedpoint.VarFxp method), 2
val() (pysnark.runtime.Var method), 5
vals() (pysnark.runtime.Var class method), 5
Var (class in pysnark.runtime), 5
VarFxp (class in pysnark.lib.fixedpoint), 2
vars() (pysnark.runtime.Var class method), 5
vc_assert_mult() (in module pysnark.runtime), 6
vc_declare_block() (in module pysnark.runtime), 6
vc_glue() (in module pysnark.runtime), 6
vc_p (in module pysnark.options), 4

Z

zero() (pysnark.runtime.Var class method), 5