Lattice-based public-key cryptosystems

D. J. Bernstein

NIST post-quantum competition: 82 submissions in first round, from hundreds of people.

- -13 submissions that NIST
- declared incomplete or improper.
- 5 withdrawn submissions.
- 3 merged submissions.

22 signature-system submissions. 5 lattice-based: Dilithium; DRS (broken); FALCON*; pqNTRUSign*; qTESLA.

47 encryption-system submissions. 20 lattice-based: Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard *****; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.

Lattice-based public-key cryptosystems

D. J. Bernstein

NIST post-quantum competition: 82 submissions in first round, from hundreds of people. -13 submissions that NIST

declared incomplete or improper.

- 5 withdrawn submissions.
- 3 merged submissions.

22 signature-system submissions. 5 lattice-based: Dilithium; DRS (broken); FALCON*; pqNTRUSign*; qTESLA.

47 encryption-system submissions. 20 lattice-based: Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard *****; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium. *****: submitter claims patent on

1

this submission. Warning: even without *****, submission could be covered by other patents!

- based
- ey cryptosystems
- rnstein
- st-quantum competition:
- issions in first round,
- ndreds of people.
- bmissions that NIST
- incomplete or improper.
- ndrawn submissions.
- ged submissions.
- ture-system submissions. -based: Dilithium; oken); FALCON*; JSign 😭; qTESLA.

- 47 encryption-system submissions. 20 lattice-based: Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard *****; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.
- *****: submitter claims patent on this submission. Warning: even without *****, submission could be covered by other patents!



First ser encrypti Hoffstein Annound

at Crypt Patenteo ystems

- m competition:
- first round,
- people.
- that NIST
- te or improper.
- omissions.
- issions.
- m submissions.
- ilithium;
- LCON *;
- TESLA.

47 encryption-system submissions. 20 lattice-based: Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard*; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.

Submitter claims patent on this submission. Warning: even without 2, submission could be covered by other patents!

First serious lattice encryption system Hoffstein–Pipher–S Announced 20 Aug

at Crypto 1996 ru Patented until 201 ition: I, 1

oper.

ions.

47 encryption-system submissions. 20 lattice-based: Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard*; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.

Submitter claims patent on this submission. Warning: even without 2, submission could be covered by other patents! First serious lattice-based encryption system: NTRU f Hoffstein–Pipher–Silverman. Announced 20 August 1996 at Crypto 1996 rump session Patented until 2017.

47 encryption-system submissions. 20 lattice-based:

Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard *****; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.

*****: submitter claims patent on this submission. Warning: even without *****, submission could be covered by other patents!

2

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman. Announced 20 August 1996 at Crypto 1996 rump session.

Patented until 2017.

47 encryption-system submissions. 20 lattice-based:

Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard *****; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.

*****: submitter claims patent on this submission. Warning: even without *****, submission could be covered by other patents!

2

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

47 encryption-system submissions. 20 lattice-based:

Compact LWE (broken); Ding *****; EMBLEM; Frodo; HILA5 (CCA broken); KCL*; KINDI; Kyber; LAC; LIMA; Lizard *****; LOTUS; NewHope; NTRUEncrypt; NTRU HRSS; NTRU Prime; Odd Manhattan; Round2^{*}; SABER; Titanium.

*****: submitter claims patent on this submission. Warning: even without *****, submission could be covered by other patents!

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

2

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

ption-system submissions. e-based:

2

t LWE t (broken); EMBLEM; Frodo; CCA broken); KCL*; Kyber; LAC; LIMA; ; LOTUS; NewHope; ncrypt; NTRU HRSS; Prime; Odd Manhattan; **;** SABER; Titanium.

nitter claims patent on mission. Warning: even *****, submission could be by other patents!

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

1996 pa attack p problem applied to attac

em submissions.

2

(broken); l; Frodo; en); KCL*; C; LIMA; NewHope;

rru hrss;

d Manhattan;

R; Titanium.

ms patent on Varning: even ssion could be patents! First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

1996 paper conver attack problem int problem (suboptin applied LLL (not s to attack the lattic

n	
\mathbf{O}	15.
	or

;

e; 5;

an;

n.

on

ven

d be

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and applied LLL (not state of th to attack the lattice problem

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

3

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

3

Coppersmith–Shamir, Eurocrypt 1997: better conversion +better attacks than LLL. Quantitative impact? Unclear.

First serious lattice-based encryption system: NTRU from Hoffstein–Pipher–Silverman.

Announced 20 August 1996 at Crypto 1996 rump session. Patented until 2017.

First version of NTRU paper, handed out at Crypto 1996, finally put online in 2016: web.securityinnovation.com /hubfs/files/ntru-orig.pdf

Proposed 104-byte public keys for 2^{80} security.

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

3

Coppersmith–Shamir, Eurocrypt 1997: better conversion +better attacks than LLL. Quantitative impact? Unclear.

NTRU paper, ANTS 1998: proposed 147-byte or 503-byte keys for 2^{77} or 2^{170} security.

ious lattice-based on system: NTRU from n–Pipher–Silverman. 3

ced 20 August 1996 to 1996 rump session. d until 2017.

sion of NTRU paper, out at Crypto 1996, ut online in 2016: curityinnovation.com

files/ntru-orig.pdf

d 104-byte public keys security. 1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

Coppersmith–Shamir, Eurocrypt 1997: better conversion + better attacks than LLL. Quantitative impact? Unclear.

NTRU paper, ANTS 1998: proposed 147-byte or 503-byte keys for 2^{77} or 2^{170} security.

Let's try Debian: Fedora: Source: Web: sa Sage is + many + a few sage: 1 1000000 sage: f

- 31721350
- sage:

e-based

: NTRU from Silverman. 3

gust 1996 mp session.

.7.

ΓRU paper,

pto 1996,

n 2016:

novation.com

tru-orig.pdf

e public keys

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

Coppersmith–Shamir, Eurocrypt 1997: better conversion + better attacks than LLL. Quantitative impact? Unclear.

NTRU paper, ANTS 1998: proposed 147-byte or 503-byte keys for 2⁷⁷ or 2¹⁷⁰ security.

Let's try NTRU or

Debian: apt inst

Fedora: yum inst

Source: www.sage

Web: sagecell.

Sage is Python 2 + many math libra + a few syntax dif

sage: 10^6 # pow
1000000

sage: factor(314

317213509 * 9903

sage:

rom

3

۱.

r,

.com

.pdf

eys

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

Coppersmith–Shamir, Eurocrypt 1997: better conversion +better attacks than LLL. Quantitative impact? Unclear.

NTRU paper, ANTS 1998: proposed 147-byte or 503-byte keys for 2^{77} or 2^{170} security.

4

1000000

sage:

Let's try NTRU on the com

- Debian: apt install sage
- Fedora: yum install sage
- Source: www.sagemath.org
- Web: sagecell.sagemath
- Sage is Python 2
- + many math libraries
- + a few syntax differences:
- sage: 10^6 # power, not x
- sage: factor(314159265358
- 317213509 * 990371647

1996 paper converted NTRU attack problem into a lattice problem (suboptimally), and then applied LLL (not state of the art) to attack the lattice problem.

Coppersmith–Shamir, Eurocrypt 1997: better conversion +better attacks than LLL. Quantitative impact? Unclear.

NTRU paper, ANTS 1998: proposed 147-byte or 503-byte keys for 2^{77} or 2^{170} security.

Let's try NTRU on the computer.

4

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2

+ many math libraries

+ a few syntax differences:

sage: 10^6 # power, not xor 1000000

317213509 * 990371647

sage:

- sage: factor(314159265358979323)

per converted NTRU roblem into a lattice (suboptimally), and then LLL (not state of the art) k the lattice problem.

mith-Shamir, Eurocrypt

- etter conversion +
- ttacks than LLL.
- ative impact? Unclear.

paper, ANTS 1998: 147-byte or 503-byte 2^{77} or 2^{170} security.

Let's try NTRU on the computer.

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org

Web: sagecell.sagemath.org

Sage is Python 2

- + many math libraries
- + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

4

sage: Z: sage: # sage: # sage: # sage:

rted NTRU to a lattice hally), and then state of the art) ce problem. 4

mir, Eurocrypt ersion +

n LLL.

ct? Unclear.

TS 1998:

or 503-byte ⁰ security. Let's try NTRU on the computer.

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2

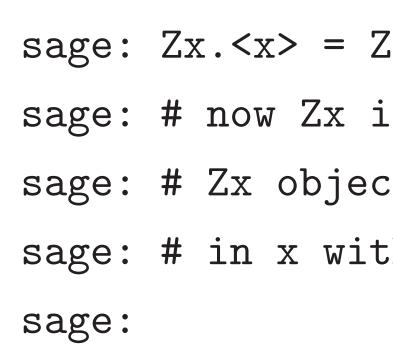
+ many math libraries

+ a few syntax differences:

sage: 10^6 # power, not xor
1000000

sage: factor(314159265358979323)
317213509 * 990371647

sage:



```
then
e art)
٦.
```

rypt

ar.

/te

Let's try NTRU on the computer. Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org Sage is Python 2 + many math libraries sage: factor(314159265358979323)

sage:

5

+ a few syntax differences: sage: 10^6 # power, not xor 1000000 317213509 * 990371647

sage:

sage: $Zx. \langle x \rangle = ZZ[]$

- sage: # now Zx is a class
- sage: # Zx objects are pc
- sage: # in x with int coe

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2 + many math libraries + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

```
sage: Zx. \langle x \rangle = ZZ[]
sage: # now Zx is a class
sage: # Zx objects are polys
sage: # in x with int coeffs
sage:
```

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2 + many math libraries + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage:

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2 + many math libraries + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage: f $4*x^2 + x + 3$ sage:

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2 + many math libraries + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage: f $4*x^2 + x + 3$ sage: g = Zx([2,7,1])sage:

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2 + many math libraries + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage: f $4*x^2 + x + 3$ sage: g = Zx([2,7,1])sage: g $x^2 + 7 + x + 2$ sage:

Debian: apt install sagemath Fedora: yum install sagemath Source: www.sagemath.org Web: sagecell.sagemath.org

Sage is Python 2 + many math libraries + a few syntax differences:

sage: 10^6 # power, not xor 1000000

sage: factor(314159265358979323)

317213509 * 990371647

sage:

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage: f $4*x^2 + x + 3$ sage: g = Zx([2,7,1])sage: g $x^2 + 7 + x + 2$ sage: f+g # built-in add $5*x^2 + 8*x + 5$ sage:

NTRU on the computer.

5

apt install sagemath
yum install sagemath
www.sagemath.org
agecell.sagemath.org

- Python 2
- math libraries
- syntax differences:

0^6 # power, not xor

actor(314159265358979323)09 * 990371647

sage:	Zx. <x> = ZZ[]</x>
sage:	<pre># now Zx is a class</pre>
sage:	# Zx objects are p
sage:	# in x with int co
sage:	f = Zx([3,1,4])
sage:	f
4*x^2	+ x + 3
sage:	g = Zx([2,7,1])
sage:	g
x^2 +	7*x + 2
sage:	f+g # built-in a
5*x^2	+ 8*x + 5
sage:	

ss oolys oeffs

add

- sage: f: 4*x^3 +
- sage:

n the computer.

5

all sagemath all sagemath emath.org sagemath.org

aries fferences:

er, not xor

159265358979323) 71647

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage: f $4*x^2 + x + 3$ sage: g = Zx([2,7,1])sage: g $x^2 + 7*x + 2$ sage: f+g # built-in add $5*x^2 + 8*x + 5$ sage:

sage: f*x # bu 4*x^3 + x^2 + 3* sage:

5		6	
puter.	sage: $Zx. < x > = ZZ[]$		sage:
math	sage: # now Zx is a class		4*x^3
math	sage: # Zx objects are polys		sage:
	<pre>sage: # in x with int coeffs</pre>		
.org	sage: $f = Zx([3,1,4])$		
• • • • 8	sage: f		
	$4*x^2 + x + 3$		
	sage: $g = Zx([2,7,1])$		
	sage: g		
or	$x^2 + 7*x + 2$		
	<pre>sage: f+g # built-in add</pre>		
979323)	5*x^2 + 8*x + 5		
	sage:		

f*x # built-in mu + x^2 + 3*x

sage:	Zx. <x> = ZZ[]</x>
sage:	<pre># now Zx is a class</pre>
sage:	# Zx objects are polys
sage:	# in x with int coeffs
sage:	f = Zx([3,1,4])
sage:	f
4*x^2	+ x + 3
sage:	g = Zx([2,7,1])
sage:	g
x^2 +	7*x + 2
sage:	f+g # built-in add
5*x^2	+ 8*x + 5
sage:	

sage:	f۶	κX		#	່bເ
4*x^3	+	Х́	^2	+	3>
sage:					

uilt-in mul

7

*X

sage: $Zx. \langle x \rangle = ZZ[]$ sage: # now Zx is a class sage: # Zx objects are polys sage: # in x with int coeffs sage: f = Zx([3,1,4])sage: f $4*x^2 + x + 3$ sage: g = Zx([2,7,1])sage: g $x^2 + 7*x + 2$ sage: f+g # built-in add $5*x^2 + 8*x + 5$ sage:

sage: f*x # built-in mul $4*x^3 + x^2 + 3*x$ sage: f*x^2 $4*x^4 + x^3 + 3*x^2$ sage:

sage:	Zx. < x > = ZZ[]
sage:	<pre># now Zx is a class</pre>
sage:	# Zx objects are polys
sage:	# in x with int coeffs
sage:	f = Zx([3,1,4])
sage:	f
4*x^2	+ x + 3
sage:	g = Zx([2,7,1])
sage:	g
x^2 +	7*x + 2
sage:	f+g # built-in add
5*x^2	+ 8*x + 5
sage:	

sage:	f*x		#	b
4*x^3	+ x	^2	+	3:
sage:	f*x	^2		
4*x^4	+ x	^3	+	3:
sage:	f*2			
8*x^2	+ 2	*X	+	6
sage:				

uilt-in mul

*X

*x^2

sage:	Zx. <x> = ZZ[]</x>
sage:	# now Zx is a class
sage:	# Zx objects are polys
sage:	# in x with int coeffs
sage:	f = Zx([3,1,4])
sage:	f
4*x^2	+ x + 3
sage:	g = Zx([2,7,1])
sage:	g
x^2 +	7*x + 2
sage:	f+g # built-in add
5*x^2	+ 8*x + 5
sage:	

sage:	f۶	۴X		#	b
4*x^3	+	x	2	+	3
sage:	f۶	×χ	2		
4*x^4	+	x	`3	+	3
sage:	f۶	×2			
8*x^2	+	2*	×Χ	+	6
sage:	f۶	k(7	7*7	K)	
28*x^3	3 -	+ 7	7*7	x^2	2
sage:					

ouilt-in mul

7

*X

8*x^2

+ 21*x

sage:	Zx. <x> = ZZ[]</x>
sage:	<pre># now Zx is a class</pre>
sage:	# Zx objects are polys
sage:	# in x with int coeffs
sage:	f = Zx([3,1,4])
sage:	f
4*x^2	+ x + 3
sage:	g = Zx([2,7,1])
sage:	g
x^2 +	7*x + 2
sage:	f+g # built-in add
5*x^2	+ 8*x + 5
sage:	

sage:	f۱	κX		#	b
4*x^3	+	x	2	+	3
sage:	f۱	×χ	2`		
4*x^4	+	x	`3	+	3
sage:	f۱	×2			
8*x^2	+	2*	<x< td=""><td>+</td><td>6</td></x<>	+	6
sage:	f۱	k (7	″*∑	()	
28*x^3	3 -	- 7	′ *∑	2^2	2
sage:	f۱	×g			
4*x^4	+	29)*]	x^3	3
+ 6					
sage:					

ouilt-in mul

7

*X

*x^2

+ 21*x

+ 18*x^2 + 23*x

sage:	Zx. <x> = ZZ[]</x>
sage:	# now Zx is a class
sage:	# Zx objects are polys
sage:	# in x with int coeffs
sage:	f = Zx([3,1,4])
sage:	f
4*x^2	+ x + 3
sage:	g = Zx([2,7,1])
sage:	g
x^2 +	7*x + 2
sage:	f+g # built-in add
5*x^2	+ 8*x + 5
sage:	

sage:	f۶	κX		#	b
4*x^3	+	x	2	+	3:
sage:	f۶	κχ	2		
4*x^4	+	x	`3	+	3:
sage:	f۶	⊧2			
8*x^2	+	2>	kΧ	+	6
sage:	f۶	k (7	7*3	<)	
28*x^3	3 -	+ 7	7*3	ς^2	2.
28*x^3 sage:			7*3	<^2	2 -
	f۶	۴g			
sage:	f۶	۴g			
sage: 4*x^4	f> +	kg 29)*3	₹^3	3 .
sage: 4*x^4 + 6	f> +	kg 29)*3	₹^3	3 .
sage: 4*x^4 + 6 sage:	f> +	kg 29)*3	₹^3	3 .

uilt-in mul

7

*X

*x^2

+ 21*x

+ 18*x^2 + 23*x

2+f*(7*x)+f*x^2

x. < x > = ZZ[]now Zx is a class Zx objects are polys in x with int coeffs = Zx([3,1,4])

6

x + 3 = Zx([2,7,1])

*x + 2

+g # built-in add 8*x + 5

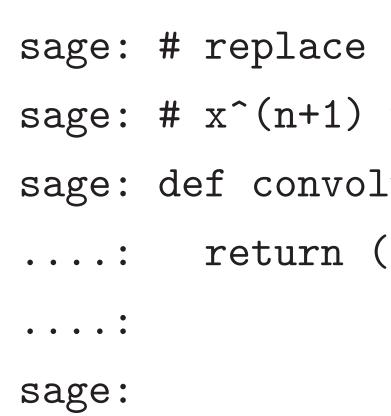
sage: f*x # built-in mul $4*x^3 + x^2 + 3*x$ sage: f*x^2 $4*x^4 + x^3 + 3*x^2$ sage: f*2 8*x² + 2*x + 6 sage: f*(7*x) $28 \times 3 + 7 \times 2 + 21 \times 1$ sage: f*g $4*x^4 + 29*x^3 + 18*x^2 + 23*x$ + 6 sage: $f*g == f*2+f*(7*x)+f*x^2$ True sage:

sage: # sage: # sage: de • • • • • • • • • •

7

sage:

6	
Z[]	<pre>sage: f*x # built-in mul</pre>
s a class	$4*x^3 + x^2 + 3*x$
ts are polys	<pre>sage: f*x^2</pre>
h int coeffs	$4*x^4 + x^3 + 3*x^2$
1,4])	<pre>sage: f*2</pre>
	8*x^2 + 2*x + 6
	sage: f*(7*x)
7,1])	28*x^3 + 7*x^2 + 21*x
	sage: f*g
	4*x^4 + 29*x^3 + 18*x^2 + 23*x
uilt-in add	+ 6
	<pre>sage: f*g == f*2+f*(7*x)+f*x^2</pre>
	True
	sage:



	6	7		
	<pre>sage: f*x # built-in mul</pre>		sage:	
	$4*x^3 + x^2 + 3*x$		sage:	•
lys	<pre>sage: f*x^2</pre>		sage:	
ffs	$4*x^4 + x^3 + 3*x^2$		• • • • •	
	sage: f*2		• • • • •	
	8*x^2 + 2*x + 6		sage:	
	sage: $f*(7*x)$			
	28*x^3 + 7*x^2 + 21*x			
	<pre>sage: f*g</pre>			
	4*x^4 + 29*x^3 + 18*x^2 + 23*x			
.dd	+ 6			
	<pre>sage: f*g == f*2+f*(7*x)+f*x^2</pre>			
	True			
	sage:			

- # replace xîn with
- # x^(n+1) with x, e
- def convolution(f,g
 - return (f*g) % (x

sage:	f*x	#	built-in mul
4*x^3	+ x^2	+	3*x
sage:	$f*x^2$		
4*x^4	+ x^3	+	3*x^2
sage:	f*2		
8*x^2	+ 2*x	+	6
sage:	f*(7*>	()	
28*x^3	3 + 7*3	ζ^2	2 + 21*x
sage:	f*g		
4*x^4	+ 29*3	x^ 3	3 + 18*x^2 + 23*x
+ 6			
sage:	f*g ==	= 1	f*2+f*(7*x)+f*x^2
True			
sage:			

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g): • • • • • • • • • • sage:

7

return (f*g) % (x^n-1)

sage: f	*X	#	built-in	mul
4*x^3 +	· x^2	+	3*x	
sage: f	*x^2			
4*x^4 +	· x^3	+	3*x^2	
sage: f	*2			
8*x^2 +	· 2*x	+	6	
sage: f	*(7*x	:)		
28*x^3	+ 7*x	<u> </u>	2 + 21*x	
sage: f	*g			
4*x^4 +	· 29*x	<u>^</u> 3	8 + 18*x^2	+ 23*x
+ 6				
sage: f	*g ==	f =	f*2+f*(7*x)+f*x^2
True				
sage:				

sage: # replace x^n with 1, sage: # x^(n+1) with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1): sage: n = 3 # global variable sage:

sage:	f*x	#	built-in	mul
4*x^3	+ x^2	+	3*x	
sage:	f*x^2			
4*x^4	+ x^3	+	3*x^2	
sage:	f*2			
8*x^2	+ 2*x	+	6	
sage:	f*(7*x)	z)		
28*x^3	3 + 7*x	c^2	2 + 21*x	
sage:	f*g			
4*x^4	+ 29*x	c^3	8 + 18*x^2	2 + 23*x
+ 6				
sage:	f*g ==	= f	f*2+f*(7*z	x)+f*x^2
True				
sage:				

sage: # replace x^n with 1, sage: # x^(n+1) with x, etc. sage: def convolution(f,g): return (f*g) % (x^n-1) • • • • • • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3*x + 4$ sage:

sage:	f*x	#	built-in	mul
4*x^3	+ x^2	+	3*x	
sage:	f*x^2			
4*x^4	+ x^3	+	3*x^2	
sage:	f*2			
8*x^2	+ 2*x	+	6	
sage:	f*(7*x)	()		
28*x^3	3 + 7*x	x^2	2 + 21*x	
sage:	f*g			
4*x^4	+ 29*x	r^3	8 + 18*x^2	2 + 23*x
+ 6				
sage:	f*g ==	= f	f*2+f*(7*x	x)+f*x^2
True				
sage:				

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage:

<pre>sage: f*x # built-in mul</pre>
$4*x^3 + x^2 + 3*x$
<pre>sage: f*x^2</pre>
$4*x^4 + x^3 + 3*x^2$
sage: f*2
8*x^2 + 2*x + 6
sage: f*(7*x)
28*x^3 + 7*x^2 + 21*x
<pre>sage: f*g</pre>
$4*x^4 + 29*x^3 + 18*x^2 + 23*x$
+ 6
sage: $f*g == f*2+f*(7*x)+f*x^2$
True
sage:

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$ sage:

8

*X	#	built-in mul	
x^2	+	3*x	
*x^2			
x^3	+	3*x^2	
*2			
2*x	+	6	
*(7*3	<)		
+ 7*3	۲^2	2 + 21*x	
*g			
29*3	x^3	3 + 18*x^2 + 23*z	ζ
*g ==	= 1	f*2+f*(7*x)+f*x^2	2
1			

7

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$ sage:

sage: de

- • • • • • • • • • • • • • • • • •
- sage:

7	
ilt-in mul	<pre>sage: # replace x^n with 1,</pre>
X	<pre>sage: # x^(n+1) with x, etc.</pre>
	<pre>sage: def convolution(f,g):</pre>
x^2	: return (f*g) % (x^n-1)
	• • • • •
	<pre>sage: n = 3 # global variable</pre>
	<pre>sage: convolution(f,x)</pre>
21*x	$x^2 + 3*x + 4$
	<pre>sage: convolution(f,x^2)</pre>
18*x^2 + 23*x	$3*x^2 + 4*x + 1$
	<pre>sage: convolution(f,g)</pre>
+f*(7*x)+f*x^2	18*x^2 + 27*x + 35
	sage:

sage: def random: f = list: for j: return Z: sage:

7	8	
.1	<pre>sage: # replace x^n with 1,</pre>	sage: c
	<pre>sage: # x^(n+1) with x, etc.</pre>	• • • • •
	<pre>sage: def convolution(f,g):</pre>	• • • • •
	: return (f*g) % (x^n-1)	• • • •
		• • • • •
	<pre>sage: n = 3 # global variable</pre>	sage:
	<pre>sage: convolution(f,x)</pre>	
	$x^2 + 3 x + 4$	
	<pre>sage: convolution(f,x^2)</pre>	
23*x	$3*x^2 + 4*x + 1$	
	<pre>sage: convolution(f,g)</pre>	
f*x^2	18*x^2 + 27*x + 35	
	sage:	

def randompoly(): f = list(randrang for j in range(return Zx(f)

sage: # replace x^n with 1, sage: # x^(n+1) with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$ sage:

sage: def randompoly(): ...: f = list(randrange(3)-1)• \ldots : return Zx(f)• • • • • sage:

8

for j in range(n))

sage: # replace x^n with 1, sage: # x^(n+1) with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$

sage:

sage: def randompoly(): ...: f = list(randrange(3)-1)• \ldots : return Zx(f)• • • • • sage: n = 7sage:

8

for j in range(n))

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$

sage:

sage: def randompoly(): ...: f = list(randrange(3)-1)• \ldots : return Zx(f)• • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage:

8

for j in range(n))

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g):: return (f*g) % (x^n-1) • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$

sage:

sage: def randompoly(): ...: f = list(randrange(3)-1)• \ldots : return Zx(f)• • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage: randompoly() $x^6 + x^5 + x^3 - x$ sage:

8

for j in range(n))

sage: # replace x^n with 1, sage: $\# x^{(n+1)}$ with x, etc. sage: def convolution(f,g):: return $(f*g) % (x^n-1)$ • • • • • sage: n = 3 # global variable sage: convolution(f,x) $x^2 + 3 + x + 4$ sage: convolution(f,x^2) $3*x^2 + 4*x + 1$ sage: convolution(f,g) $18 \times 2 + 27 \times 35$ sage:

sage: def randompoly(): • \ldots : return Zx(f)• • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage: randompoly() $x^6 + x^5 + x^3 - x$ sage: randompoly() $-x^{6} + x^{5} + x^{4} - x^{3} - x^{2} +$ x + 1 sage:

8

...: f = list(randrange(3)-1)for j in range(n))

replace x^n with 1,
x^(n+1) with x, etc.
ef convolution(f,g):
return (f*g) % (x^n-1)
= 3 # global variable
onvolution(f,x)
*x + 4
onvolution(f,x^2)
4*x + 1
onvolution(f,g)
+ 27*x + 35

8

sage: def randompoly():: f = list(randrange(3)-1)for j in range(n)) • • • • • return Zx(f) • • • • • • • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage: randompoly() $x^6 + x^5 + x^3 - x$ sage: randompoly() $-x^6 + x^5 + x^4 - x^3 - x^2 +$ x + 1 sage:

9

Will use Some ch in subm n = 701*n* = 743 n = 761

	8	
xîn with 1,		<pre>sage: def randompoly():</pre>
with x, etc.		: $f = list(randrange(3)-1)$
ution(f,g):		<pre>: for j in range(n))</pre>
f*g) % (x^n-1)		: return Zx(f)
		• • • •
lobal variable		sage: $n = 7$
n(f,x)		<pre>sage: randompoly()</pre>
		$-x^3 - x^2 - x - 1$
n(f,x^2)		<pre>sage: randompoly()</pre>
		$x^6 + x^5 + x^3 - x$
n(f,g)		<pre>sage: randompoly()</pre>
35		$-x^6 + x^5 + x^4 - x^3 - x^2 +$
		x + 1
		sage:

Will use bigger n Some choices of n in submissions to n = 701 for NTRL n = 743 for NTRL

9

n = 761 for sntru

	8	
1,	<pre>sage: def randompoly():</pre>	Will use
tc.	: $f = list(randrange(3)-1$	Some cl
;):	<pre>: for j in range(n))</pre>	in subm
^n-1)	: return Zx(f)	
		n = 701
iable	sage: $n = 7$	<i>n</i> = 743
	<pre>sage: randompoly()</pre>	<i>n</i> = 761
	$-x^3 - x^2 - x - 1$	
	<pre>sage: randompoly()</pre>	
	$x^6 + x^5 + x^3 - x$	
	<pre>sage: randompoly()</pre>	
	$-x^6 + x^5 + x^4 - x^3 - x^2 +$	
	x + 1	
	sage:	

e bigger *n* for security

- choices of n
- nissions to NIST:
- 1 for NTRU HRSS.
- 3 for NTRUEncrypt.
- 1 for sntrup4591761

sage: def randompoly():: f = list(randrange(3)-1)for j in range(n)) • • • • •: return Zx(f) • • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage: randompoly() $x^6 + x^5 + x^3 - x$ sage: randompoly() $-x^6 + x^5 + x^4 - x^3 - x^2 +$ x + 1 sage:

Will use bigger *n* for security.

9

Some choices of *n* in submissions to NIST:

n = 701 for NTRU HRSS.

n = 743 for NTRUEncrypt.

n = 761 for sntrup4591761.

sage: def randompoly():: f = list(randrange(3)-1)for j in range(n)) • • • • •: return Zx(f) • • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage: randompoly() $x^6 + x^5 + x^3 - x$ sage: randompoly() $-x^{6} + x^{5} + x^{4} - x^{3} - x^{2} +$ x + 1 sage:

Will use bigger *n* for security. Some choices of *n* in submissions to NIST: n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761.

9

Overkill against attack algorithms known today, even for future attacker with quantum computer.

9 sage: def randompoly():: f = list(randrange(3)-1)for j in range(n)) • • • • •: return Zx(f) • • • • • sage: n = 7sage: randompoly() $-x^3 - x^2 - x - 1$ sage: randompoly() $x^6 + x^5 + x^3 - x$ sage: randompoly() $-x^{6} + x^{5} + x^{4} - x^{3} - x^{2} +$ x + 1 sage:

Will use bigger *n* for security. Some choices of *n* in submissions to NIST: n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761. known today, even for future attacker with quantum computer.

10

- Overkill against attack algorithms

Can we find better algorithms?

	9
<pre>sage: def randompoly():</pre>	
<pre>: f = list(randrange(3)-1</pre>	
<pre>: for j in range(n))</pre>	
: return Zx(f)	
• • • •	
sage: $n = 7$	
<pre>sage: randompoly()</pre>	
$-x^3 - x^2 - x - 1$	
<pre>sage: randompoly()</pre>	
$x^6 + x^5 + x^3 - x$	
<pre>sage: randompoly()</pre>	
$-x^6 + x^5 + x^4 - x^3 - x^2 +$	
x + 1	
sage:	

Will use bigger *n* for security. Some choices of *n* in submissions to NIST: n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761. known today, even for future attacker with quantum computer. Can we find better algorithms? 1998 NTRU paper took n = 503.

- Overkill against attack algorithms

ef randompoly():

f = list(randrange(3)-1)for j in range(n)) return Zx(f)

= 7

andompoly()

 $x^2 - x - 1$

andompoly()

 $5 + x^{3} - x$

andompoly()

 $x^5 + x^4 - x^3 - x^2 +$

Will use bigger *n* for security.

Some choices of *n* in submissions to NIST:

9

n = 701 for NTRU HRSS.

n = 743 for NTRUEncrypt.

n = 761 for sntrup4591761.

Overkill against attack algorithms known today, even for future attacker with quantum computer.

Can we find better algorithms?

1998 NTRU paper took n = 503.

10

Modular

For integ

Sage's "

outputs

Matches

```
poly():
(randrange(3)-1
in range(n))
x(f)
```

9

() 1 () - x

- x^3 - x^2 +

()

Will use bigger *n* for security.

Some choices of *n* in submissions to NIST:

- n = 701 for NTRU HRSS.
- n = 743 for NTRUEncrypt.
- n = 761 for sntrup4591761.

Overkill against attack algorithms known today, even for future attacker with quantum computer.

Can we find better algorithms?

1998 NTRU paper took n = 503.

Modular reduction

For integers u, q v Sage's "u%q" alwa outputs between C

Matches standard

e(3)-1n))

9

Will use bigger *n* for security. Some choices of *n* in submissions to NIST: n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761. Overkill against attack algorithms known today, even for future attacker with quantum computer. Can we find better algorithms?

1998 NTRU paper took n = 503.

10

Matches standard math defi

x^2 +

Modular reduction

For integers u, q with q > 0Sage's "u%q" always produc outputs between 0 and q -

Will use bigger *n* for security.

Some choices of *n* in submissions to NIST:

n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761.

Overkill against attack algorithms known today, even for future attacker with quantum computer.

Can we find better algorithms?

1998 NTRU paper took n = 503.

10

Modular reduction

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Will use bigger *n* for security.

Some choices of *n* in submissions to NIST:

n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761.

Overkill against attack algorithms known today, even for future attacker with quantum computer.

Can we find better algorithms?

1998 NTRU paper took n = 503.

10

Modular reduction

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Matches standard math definition.

Will use bigger *n* for security.

Some choices of *n* in submissions to NIST:

n = 701 for NTRU HRSS. n = 743 for NTRUEncrypt. n = 761 for sntrup4591761.

Overkill against attack algorithms known today, even for future attacker with quantum computer.

Can we find better algorithms?

1998 NTRU paper took n = 503.

10

Modular reduction

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.

- Matches standard math definition.

- bigger *n* for security.
- noices of *n*
- issions to NIST:
- for NTRU HRSS.
- for NTRUEncrypt.
- for sntrup4591761.
- against attack algorithms oday, even for future with quantum computer.
- find better algorithms?
- ⁻RU paper took n = 503.

10

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.

- sage: de
- sage:

- sage:
- sage:
- sage:
- sage:

for security.

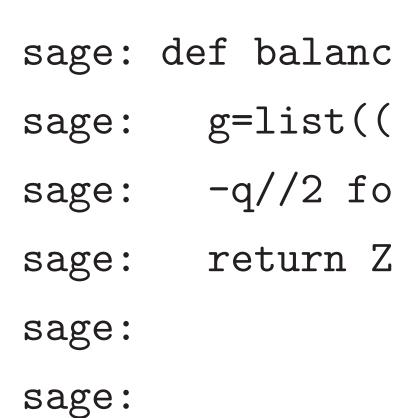
- NIST:
- J HRSS.
- JEncrypt.
- up4591761.
- tack algorithms for future
- ntum computer.
- r algorithms?
- $r \, took \, n = 503.$

10

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1. Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0 in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.



l. rithms

/.

10

- Ŋ
- puter.
- ns?
- 503.

Modular reduction

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake. sage: sage:

11

sage:

sage:

sage:

sage:

def balancedmod(f,q g=list(((f[i]+q// -q//2 for i in ra return Zx(g)

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake. sage: def balancedmod(f,q): sage: sage: return Zx(g) sage: sage: sage:

11

g=list(((f[i]+q//2)%q) -q//2 for i in range(n))

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.

```
sage: def balancedmod(f,q):
sage:
        return Zx(g)
sage:
sage:
sage: u = 314 - 159 * x
sage:
```

11

g=list(((f[i]+q//2)%q) sage: -q//2 for i in range(n))

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.

sage: def balancedmod(f,q): sage: g=list(((f[i]+q//2)%q) sage: -q//2 for i in range(n)) return Zx(g) sage: sage: sage: u = 314 - 159 * xsage: u % 200 $-159 \times x + 114$ sage:

11

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.

sage: def balancedmod(f,q): sage: g=list(((f[i]+q//2)%q) sage: -q//2 for i in range(n)) sage: return Zx(g) sage: sage: u = 314 - 159 * xsage: u % 200 -159 * x + 114sage: (u - 400) % 200 -159*x - 86 sage:

11

Modular reduction

For integers u, q with q > 0, Sage's "u%q" always produces outputs between 0 and q - 1.

Matches standard math definition.

11

Warning: Typically u < 0 produces u%q < 0in lower-level languages, so nonzero output leaks input sign.

Warning: For polynomials u, Sage can make the same mistake.

12 sage: def balancedmod(f,q): sage: g=list(((f[i]+q//2)%q) sage: -q//2 for i in range(n)) sage: return Zx(g) sage: sage: u = 314 - 159 * xsage: u % 200 -159 * x + 114sage: (u - 400) % 200 -159*x - 86 sage: balancedmod(u,200) 41*x - 86 sage:

reduction

gers u, q with q > 0, u%q" always produces between 0 and q - 1.

standard math definition.

: Typically oduces u%q < 0level languages, so output leaks input sign.

: For polynomials u,

n make the same mistake.

sage: def balancedmod(f,q): sage: g=list(((f[i]+q//2)%q) sage: -q//2 for i in range(n)) return Zx(g) sage: sage: sage: u = 314 - 159 * xsage: u % 200 $-159 \times x + 114$ sage: (u - 400) % 200 -159*x - 86 sage: balancedmod(u,200) 41*x - 86 sage:

11

12

sage: de • • • • •

- • • • • • • • • • • • • • • • • •
- sage:

-

with q > 0, and q - 1.

math definition.

11

y %q < 0 uages, so aks input sign.

nomials u,

e same mistake.

sage: def balancedmod(f,q): sage: g=list(((f[i]+q//2)%q) sage: -q//2 for i in range(n)) sage: return Zx(g) sage: sage: u = 314-159 * xsage: u % 200 -159 * x + 114sage: (u - 400) % 200 -159*x - 86 sage: balancedmod(u,200) 41*x - 86 sage:

sage:	def invert
• • • • •	Fp = Int
• • • • •	Fpx = Zx
• • • • •	T = Fpx.
• • • • •	return Z
• • • • •	
• • • • •	

sage:

11	12
	<pre>sage: def balancedmod(f,q):</pre>
	<pre>sage: g=list(((f[i]+q//2)%q)</pre>
' es	<pre>sage: -q//2 for i in range(n))</pre>
1	sage: return Zx(g)
- •	sage:
nition.	sage: u = 314-159*x
	sage: u % 200
	-159*x + 114
	sage: (u - 400) % 200
sign.	-159*x - 86
0	<pre>sage: balancedmod(u,200)</pre>
7	41*x - 86
stake.	sage:

sage: def invertmodprime(....: Fp = Integers(p): Fpx = Zx.change_r: T = Fpx.quotient(....: return Zx(lift(1/

• • • • •

sage:

L L L L L L L L L L L L L L L L L L L
<pre>sage: def balancedmod(f,q):</pre>
<pre>sage: g=list(((f[i]+q//2)%q)</pre>
<pre>sage: -q//2 for i in range(n))</pre>
<pre>sage: return Zx(g)</pre>
sage:
sage: u = 314-159*x
sage: u % 200
-159*x + 114
sage: (u - 400) % 200
-159*x - 86
<pre>sage: balancedmod(u,200)</pre>
41*x - 86
sage:

sage: def inver: Fp = Int....: Fpx = Zt....: T = Fpx....: return

sage:

12

sage: def invertmodprime(f,p):
....: Fp = Integers(p)

13

Fpx = Zx.change_ring(Fp)

 $T = Fpx.quotient(x^n-1)$

return Zx(lift(1/T(f)))

<pre>sage: def balancedmod(f,q):</pre>
<pre>sage: g=list(((f[i]+q//2)%q)</pre>
<pre>sage: -q//2 for i in range(n)</pre>
<pre>sage: return Zx(g)</pre>
sage:
sage: u = 314-159*x
sage: u % 200
-159*x + 114
sage: (u - 400) % 200
-159*x - 86
<pre>sage: balancedmod(u,200)</pre>
41*x - 86
sage:

sage:	def inver
• • • • •	Fp = In
•	Fpx = Z
• • • • •	T = Fpx
• • • • •	return
• • • • •	
sage:	n = 7
sage:	

12

)

tmodprime(f,p):
tegers(p)

13

Xx.change_ring(Fp)

 $.quotient(x^n-1)$

Zx(lift(1/T(f)))

1
<pre>sage: def balancedmod(f,q):</pre>
<pre>sage: g=list(((f[i]+q//2)%q)</pre>
<pre>sage: -q//2 for i in range(n))</pre>
<pre>sage: return Zx(g)</pre>
sage:
sage: u = 314-159*x
sage: u % 200
-159*x + 114
sage: (u - 400) % 200
-159*x - 86
<pre>sage: balancedmod(u,200)</pre>
41*x - 86
sage:

sage: def invertmodprime(f,p): Fp = Integers(p)• • • • • sage: n = 7sage: f = randompoly() sage:

12

Fpx = Zx.change_ring(Fp)

- ...: $T = Fpx.quotient(x^n-1)$
- ...: return Zx(lift(1/T(f)))

1
<pre>sage: def balancedmod(f,q):</pre>
<pre>sage: g=list(((f[i]+q//2)%q)</pre>
<pre>sage: -q//2 for i in range(n))</pre>
<pre>sage: return Zx(g)</pre>
sage:
sage: u = 314-159*x
sage: u % 200
-159*x + 114
sage: (u – 400) % 200
-159*x - 86
<pre>sage: balancedmod(u,200)</pre>
41*x - 86
sage:

sage:	def inver
• • • • •	$Fp = In^{-1}$
• • • • •	Fpx = Zz
• • • • •	T = Fpx
• • • • •	return 2
• • • • •	
sage:	n = 7
sage:	f = randor
sage:	f3 = invert
sage:	

12

tmodprime(f,p):
tegers(p)

x.change_ring(Fp)

13

- $.quotient(x^n-1)$
- Zx(lift(1/T(f)))

mpoly()
rtmodprime(f,3)

12	2
<pre>sage: def balancedmod(f,q):</pre>	
<pre>sage: g=list(((f[i]+q//2)%q)</pre>	
<pre>sage: -q//2 for i in range(n))</pre>	
<pre>sage: return Zx(g)</pre>	
sage:	
sage: u = 314-159*x	
sage: u % 200	
-159*x + 114	
sage: (u - 400) % 200	
-159*x - 86	
<pre>sage: balancedmod(u,200)</pre>	
41*x - 86	
sage:	

sage: def invertmodprime(f,p): Fp = Integers(p)....: Fpx = Zx.change_ring(Fp): $T = Fpx.quotient(x^n-1)$...: return Zx(lift(1/T(f)))• • • • • sage: n = 7sage: f = randompoly() sage: f3 = invertmodprime(f,3) sage: convolution(f,f3) $6*x^{6} + 6*x^{5} + 3*x^{4} + 3*x^{3} +$ $3*x^2 + 3*x + 4$ sage:

1
<pre>ef balancedmod(f,q):</pre>
g=list(((f[i]+q//2)%q)
-q//2 for i in range(n))
return Zx(g)
= 314 - 159 * x
% 200
+ 114
u - 400) % 200
- 86
alancedmod(u,200)
86

12

13
<pre>sage: def invertmodprime(f,p):</pre>
: Fp = Integers(p)
\dots : Fpx = Zx.change_ring(Fp)
: $T = Fpx.quotient(x^n-1)$
<pre>: return Zx(lift(1/T(f)))</pre>
• • • •
sage: $n = 7$
<pre>sage: f = randompoly()</pre>
<pre>sage: f3 = invertmodprime(f,3)</pre>
<pre>sage: convolution(f,f3)</pre>
6*x^6 + 6*x^5 + 3*x^4 + 3*x^3 +
$3*x^2 + 3*x + 4$
sage:

def inv asser g = i M = baC = cwhile r = if : g = Exercise invert Hint: Co

12		1
edmod(f,q):	sage:	<pre>def invertmodprime(f,p):</pre>
(f[i]+q//2)%q)	• • • • •	<pre>Fp = Integers(p)</pre>
r i in range(n))	• • • • •	<pre>Fpx = Zx.change_ring(Fp)</pre>
x(g)	• • • • •	$T = Fpx.quotient(x^n-1)$
	• • • • •	<pre>return Zx(lift(1/T(f)))</pre>
9*x	• • • • •	
	sage:	n = 7
	sage:	<pre>f = randompoly()</pre>
% 200	sage:	<pre>f3 = invertmodprime(f,3)</pre>
	sage:	<pre>convolution(f,f3)</pre>
d(u,200)	6*x^6	+ 6*x^5 + 3*x^4 + 3*x^3 +
	3*x^2	2 + 3 * x + 4
	sage:	

def invertmodpow

- assert q.is_po
- g = invertmodp
- M = balancedmo
- C = convolutio
- while True:
 - r = M(C(g,f))
 - if r == 1: r
 - g = M(C(g, 2-

Exercise: Figure o invertmodpowero Hint: Compare r

12	13	
):	<pre>sage: def invertmodprime(f,p):</pre>	def inv
2)%q)	: Fp = Integers(p)	asser
nge(n))	<pre>: Fpx = Zx.change_ring(Fp)</pre>	g = i:
	: $T = Fpx.quotient(x^n-1)$	M = b
	<pre>: return Zx(lift(1/T(f)))</pre>	C = c
		while
	sage: $n = 7$	r =
	<pre>sage: f = randompoly()</pre>	if
	<pre>sage: f3 = invertmodprime(f,3)</pre>	g =
	<pre>sage: convolution(f,f3)</pre>	Exercise
	6*x^6 + 6*x^5 + 3*x^4 + 3*x^3 +	invert
	$3*x^2 + 3*x + 4$	Hint: Co
	sage:	TIIIL. CO

vertmodpowerof2(f,o rt q.is_power_of(2) invertmodprime(f,2) calancedmod

- convolution
- e True:
- = M(C(g,f),q)
- r == 1: return g
- = M(C(g, 2-r), q)
- e: Figure out how modpowerof2 works Compare r to previou

	13
sage: de	ef invertmodprime(f,p):
• • • • •	<pre>Fp = Integers(p)</pre>
•	<pre>Fpx = Zx.change_ring(Fp)</pre>
• • • • •	$T = Fpx.quotient(x^n-1)$
• • • • •	<pre>return Zx(lift(1/T(f)))</pre>
• • • • •	
sage: n	= 7
sage: f	= randompoly()
sage: f3	3 = invertmodprime(f,3)
sage: co	onvolution(f,f3)
6*x^6 +	6*x^5 + 3*x^4 + 3*x^3 +
3*x^2 -	⊦ 3*x + 4
sage:	

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)Exercise: Figure out how invertmodpowerof2 works.

- Hint: Compare r to previous r.

ef invertmodprime(f,p):

13

Fp = Integers(p)

 $Fpx = Zx.change_ring(Fp)$ $T = Fpx.quotient(x^n-1)$

return Zx(lift(1/T(f)))

= 7

= randompoly()

3 = invertmodprime(f,3)

onvolution(f,f3)

 $6*x^5 + 3*x^4 + 3*x^3 +$

+ 3 * x + 4

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r.



sage: n sage: q

sage:

modprime(f,p):
egers(p)

13

.change_ring(Fp)
quotient(x^n-1)
x(lift(1/T(f)))

- poly()
- tmodprime(f,3)
- n(f,f3)
- $3*x^4 + 3*x^3 +$

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmod C = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r.

sage: n = 7sage: q = 256sage:

	14		
<pre>def invertmodpowerof2(f,q):</pre>		sage:	n
<pre>assert q.is_power_of(2)</pre>		sage:	q
<pre>g = invertmodprime(f,2)</pre>		sage:	
M = balancedmod			
C = convolution			
while True:			
r = M(C(g,f),q)			
if r == 1: return g			
g = M(C(g, 2-r), q)			
Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r.			
	<pre>assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmod C = convolution while True: r = M(C(g,f),q) if r == 1: return g g = M(C(g,2-r),q) Exercise: Figure out how invertmodpowerof2 works.</pre>	<pre>def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmod C = convolution while True: r = M(C(g,f),q) if r == 1: return g g = M(C(g,2-r),q) Exercise: Figure out how invertmodpowerof2 works.</pre>	<pre>def invertmodpowerof2(f,q): sage: assert q.is_power_of(2) sage: g = invertmodprime(f,2) sage: M = balancedmod C = convolution while True: r = M(C(g,f),q) if r == 1: return g g = M(C(g,2-r),q) Exercise: Figure out how invertmodpowerof2 works.</pre>

e: n = 7e: q = 256

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)Exercise: Figure out how

invertmodpowerof2 works.

Hint: Compare r to previous r.

sage: n = 7
sage: q = 256
sage:

14

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolution

while True:

r = M(C(g,f),q)

if r == 1: return g

g = M(C(g, 2-r), q)

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r. sage: n = 7sage: q = 256sage: f = randompoly() sage:

14

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r. sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^{6} - x^{4} + x^{2} + x - 1$ sage:

14

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r. sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^{6} - x^{4} + x^{2} + x - 1$ sage:

14

15

sage: g = invertmodpowerof2(f,q)

def invertmodpowerof2(f,q): assert q.is_power_of(2) g = invertmodprime(f,2) M = balancedmodC = convolutionwhile True: r = M(C(g,f),q)if r == 1: return g g = M(C(g, 2-r), q)

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r.

sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^6 - x^4 + x^2 + x - 1$ sage: g = invertmodpowerof2(f,q) sage: g 47*x^6 + 126*x^5 - 54*x^4 - $87*x^3 - 36*x^2 - 58*x + 61$ sage:

14

def	<pre>invertmodpowerof2(f,q):</pre>
as	sert q.is_power_of(2)
g	<pre>= invertmodprime(f,2)</pre>
М	= balancedmod
С	= convolution
wh	ile True:
	r = M(C(g,f),q)
	if r == 1: return g
	g = M(C(g, 2-r), q)
Evor	reiser Eigure out bow

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r.

sage: 1	n =	7
sage: d	q =	256
sage: :	f =	rando
sage: :	f	
-x^6 -	x^4	1 + x^2
sage: g	r =	inver
sage: g	50	
47*x^6	+ 1	L26*x^
87*x^3	3 –	36*x^
sage: o	con	voluti
-256*x	^5 -	- 256*:
sage:		

14

15

mpoly()

2 + x - 1tmodpowerof2(f,q)

- 5 54*x^4 -
- $2 58 \times x + 61$
- on(f,g)
- $x^4 + 256 * x + 257$

def :	<pre>invertmodpowerof2(f,q):</pre>
as	<pre>sert q.is_power_of(2)</pre>
g =	<pre>= invertmodprime(f,2)</pre>
M =	= balancedmod
C =	<pre>= convolution</pre>
wh	ile True:
]	r = M(C(g,f),q)
:	if r == 1: return g
Į	g = M(C(g, 2-r), q)
Ever	cica. Eigura aut baw

Exercise: Figure out how invertmodpowerof2 works. Hint: Compare r to previous r.

sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^{6} - x^{4} + x^{2} + x - 1$ sage: g $47*x^6 + 126*x^5 - 54*x^4 87*x^3 - 36*x^2 - 58*x + 61$ sage: convolution(f,g) sage: balancedmod(_,q) 1 sage:

14

15

sage: g = invertmodpowerof2(f,q)

- $-256*x^5 256*x^4 + 256*x + 257$

<pre>ertmodpowerof2(f,q):</pre>
t q.is_power_of(2)
nvertmodprime(f,2)
alancedmod
onvolution
True:
M(C(g,f),q)
r == 1: return g
M(C(g,2-r),q)
: Figure out how
nodpowerof2 works.
ompare r to previous r.

14

sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^{6} - x^{4} + x^{2} + x - 1$ sage: g = invertmodpowerof2(f,q) sage: g $47 \times 6 + 126 \times 5 - 54 \times 4 87*x^3 - 36*x^2 - 58*x + 61$ sage: convolution(f,g) $-256*x^5 - 256*x^4 + 256*x + 257$ sage: balancedmod(_,q) 1 sage:

15

NTRU k

Paramet n, positi q, power

14 15 sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^{6} - x^{4} + x^{2} + x - 1$ sage: g = invertmodpowerof2(f,q) sage: g 47*x^6 + 126*x^5 - 54*x^4 -87*x³ - 36*x² - 58*x + 61 sage: convolution(f,g) $-256*x^5 - 256*x^4 + 256*x + 257$ sage: balancedmod(_,q) 1 sage:

erof2(f,q):

 $wer_of(2)$

rime(f,2)

d

n

,q)

eturn g

r),q)

ut how

of2 works.

to previous r.

NTRU key genera

Parameters:

n, positive integer

q, power of 2 (e.g

):

14

```
15
sage: n = 7
sage: q = 256
sage: f = randompoly()
sage: f
-x^6 - x^4 + x^2 + x - 1
sage: g = invertmodpowerof2(f,q)
sage: g
47*x^6 + 126*x^5 - 54*x^4 -
 87*x<sup>3</sup> - 36*x<sup>2</sup> - 58*x + 61
sage: convolution(f,g)
-256*x^5 - 256*x^4 + 256*x + 257
sage: balancedmod(_,q)
1
sage:
```

Parameters: n, positive integer (e.g., 701

q, power of 2 (e.g., 4096).

sr.

NTRU key generation

15 sage: n = 7sage: q = 256sage: f = randompoly() sage: f $-x^{6} - x^{4} + x^{2} + x - 1$ sage: g = invertmodpowerof2(f,q) sage: g 47*x^6 + 126*x^5 - 54*x^4 - $87*x^3 - 36*x^2 - 58*x + 61$ sage: convolution(f,g) $-256*x^5 - 256*x^4 + 256*x + 257$ sage: balancedmod(_,q) 1 sage:

NTRU key generation

Parameters: n, positive integer (e.g., 701); q, power of 2 (e.g., 4096).

15
sage: $n = 7$
sage: q = 256
<pre>sage: f = randompoly()</pre>
sage: f
$-x^{6} - x^{4} + x^{2} + x - 1$
<pre>sage: g = invertmodpowerof2(f,q)</pre>
sage: g
47*x^6 + 126*x^5 - 54*x^4 -
87*x^3 - 36*x^2 - 58*x + 61
<pre>sage: convolution(f,g)</pre>
-256*x^5 - 256*x^4 + 256*x + 257
<pre>sage: balancedmod(_,q)</pre>
1
sage:

NTRU key generation

Parameters: *n*, positive integer (e.g., 701); q, power of 2 (e.g., 4096). Secret key: random *n*-coeff polynomial *a*; random *n*-coeff polynomial *d*; all coefficients in $\{-1, 0, 1\}$.

15
sage: $n = 7$
sage: q = 256
<pre>sage: f = randompoly()</pre>
sage: f
$-x^6 - x^4 + x^2 + x - 1$
<pre>sage: g = invertmodpowerof2(f,q)</pre>
sage: g
47*x^6 + 126*x^5 - 54*x^4 -
87*x^3 - 36*x^2 - 58*x + 61
<pre>sage: convolution(f,g)</pre>
-256*x^5 - 256*x^4 + 256*x + 257
<pre>sage: balancedmod(_,q)</pre>
1

sage:

NTRU key generation

Parameters: *n*, positive integer (e.g., 701); q, power of 2 (e.g., 4096). Secret key: random *n*-coeff polynomial *a*; random *n*-coeff polynomial *d*; all coefficients in $\{-1, 0, 1\}$. Require d invertible mod q.

Require d invertible mod 3.

T
sage: $n = 7$
sage: q = 256
<pre>sage: f = randompoly()</pre>
sage: f
$-x^{6} - x^{4} + x^{2} + x - 1$
<pre>sage: g = invertmodpowerof2(f,q)</pre>
sage: g
47*x^6 + 126*x^5 - 54*x^4 -
87*x^3 - 36*x^2 - 58*x + 61
<pre>sage: convolution(f,g)</pre>
-256*x^5 - 256*x^4 + 256*x + 257
<pre>sage: balancedmod(_,q)</pre>
1
sage:

NTRU key generation

15

Parameters: *n*, positive integer (e.g., 701); q, power of 2 (e.g., 4096). Secret key: random *n*-coeff polynomial *a*; random *n*-coeff polynomial *d*; all coefficients in $\{-1, 0, 1\}$.

Require d invertible mod q. Require d invertible mod 3.

Public key: A = 3a/d in the ring $R_q = ({\bf Z}/q)[x]/(x^n - 1).$

= 7 = 256 = randompoly() $x^4 + x^2 + x - 1$ = invertmodpowerof2(f,q) + 126*x^5 - 54*x^4 -- 36*x^2 - 58*x + 61 onvolution(f,g) $5 - 256 \times ^4 + 256 \times + 257$

alancedmod(_,q)

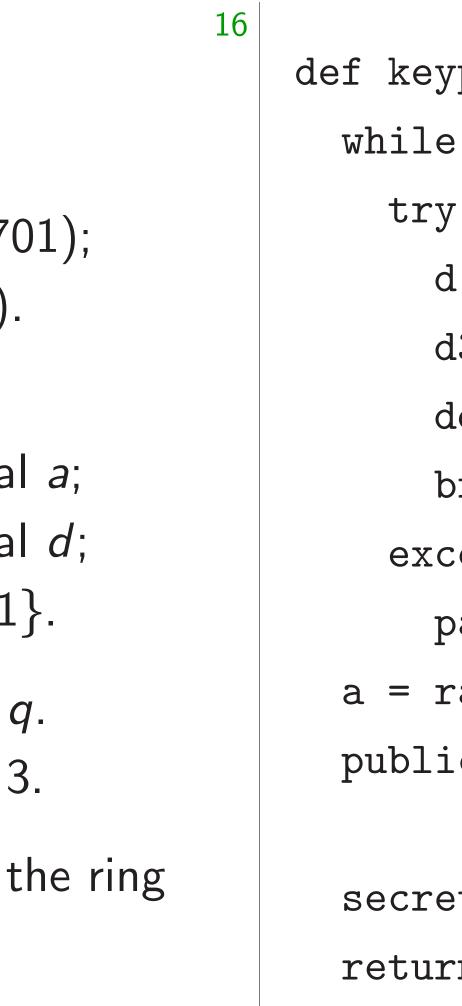
NTRU key generation

15

Parameters: *n*, positive integer (e.g., 701); q, power of 2 (e.g., 4096). Secret key: random *n*-coeff polynomial *a*; random *n*-coeff polynomial *d*; all coefficients in $\{-1, 0, 1\}$.

Require d invertible mod q. Require *d* invertible mod 3.

Public key: A = 3a/d in the ring $R_q = ({\bf Z}/q)[x]/(x^n - 1).$



d

d

d

b

p

15

poly()

+ x - 1

modpowerof2(f,q)

- 54*x^4 -

- 58*x + 61

n(f,g)

^4 + 256*x + 257

d(_,q)

NTRU key generation

Parameters:

n, positive integer (e.g., 701);*q*, power of 2 (e.g., 4096).

Secret key:

random *n*-coeff polynomial *a*; random *n*-coeff polynomial *d*; all coefficients in $\{-1, 0, 1\}$.

Require d invertible mod q. Require d invertible mod 3.

Public key: A = 3a/d in the ring $R_q = (\mathbf{Z}/q)[x]/(x^n - 1).$

def keypair(): while True: try: d = randomd3 = inverdq = inverbreak except: pass a = randompoly publickey = ba con secretkey = d, return publick

NTRU key generationdeParameters:n, positive integer (e.g., 701);q, power of 2 (e.g., 4096).	ef key while try d
n, positive integer (e.g., 701);	try d
n, positive integer (e.g., 701);	Ċ
	Ċ
9, power of 2 (e.g., 1030).	
f2(f,q) Secret key:	d
random <i>n</i> -coeff polynomial <i>a</i> ;	b
_ random <i>n</i> -coeff polynomial <i>d</i> ;	exc
61 all coefficients in $\{-1, 0, 1\}$.	p
Require <i>d</i> invertible mod <i>q</i> .	a = r
x + 257 Require <i>d</i> invertible mod 3.	publi
Public key: $A = 3a/d$ in the ring	secre
$R_q = (\mathbf{Z}/q)[x]/(x^n - 1).$	retur

- ypair():
- e True:
- y:
- d = randompoly()
- d3 = invertmodprime
- dq = invertmodpower
- break
- cept:
- pass
- randompoly()
- ickey = balancedmod
 - convolution(
- etkey = d, d3
- rn publickey,secret

NTRU key generation

Parameters:

n, positive integer (e.g., 701);

q, power of 2 (e.g., 4096).

Secret key: random *n*-coeff polynomial *a*; random *n*-coeff polynomial *d*; all coefficients in $\{-1, 0, 1\}$.

Require d invertible mod q. Require *d* invertible mod 3.

Public key: A = 3a/d in the ring $R_q = ({\bf Z}/q)[x]/(x^n - 1).$

def keypair(): while True: try: d = randompoly() break except: pass a = randompoly() secretkey = d, d3return publickey, secretkey

16

d3 = invertmodprime(d,3) dq = invertmodpowerof2(d,q)

publickey = balancedmod(3 * convolution(a,dq),q)

ey generation

ers:

ve integer (e.g., 701); r of 2 (e.g., 4096).

ey:

n-coeff polynomial *a*; *n*-coeff polynomial *d*; cients in $\{-1, 0, 1\}$.

d invertible mod q. d invertible mod 3.

ey: A = 3a/d in the ring $(x^{n} - 1)$.

def keypair(): while True: try: d = randompoly() d3 = invertmodprime(d,3) dq = invertmodpowerof2(d,q) break except: pass a = randompoly() publickey = balancedmod(3 * convolution(a,dq),q) secretkey = d, d3return publickey, secretkey

16

sage: A

sage:

tion (e.g., 701); ., 4096). olynomial *a*; olynomial d; $\{-1, 0, 1\}.$ le mod q. le mod 3. Ba/d in the ring (n - 1).

16

17 def keypair(): while True: try: d = randompoly() d3 = invertmodprime(d,3) dq = invertmodpowerof2(d,q) break except: pass a = randompoly() publickey = balancedmod(3 * convolution(a,dq),q) secretkey = d, d3return publickey, secretkey

sage: A,secretke sage:

16 17 def keypair(): while True: sage: try:); d = randompoly() d3 = invertmodprime(d,3) dq = invertmodpowerof2(d,q) 3; break *d*; except: pass a = randompoly() publickey = balancedmod(3 * convolution(a,dq),q) e ring secretkey = d, d3return publickey, secretkey

sage: A,secretkey = keypa

```
17
def keypair():
  while True:
    try:
      d = randompoly()
      d3 = invertmodprime(d,3)
      dq = invertmodpowerof2(d,q)
      break
    except:
      pass
  a = randompoly()
  publickey = balancedmod(3 *
             convolution(a,dq),q)
  secretkey = d, d3
  return publickey, secretkey
```

sage: A,secretkey = keypair() sage:

```
17
def keypair():
  while True:
    try:
      d = randompoly()
      d3 = invertmodprime(d,3)
      dq = invertmodpowerof2(d,q)
      break
    except:
      pass
  a = randompoly()
  publickey = balancedmod(3 *
             convolution(a,dq),q)
  secretkey = d, d3
  return publickey, secretkey
```

sage: A, secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage:

17
lef keypair():
while True:
try:
<pre>d = randompoly()</pre>
d3 = invertmodprime(d,3)
<pre>dq = invertmodpowerof2(d,q)</pre>
break
except:
pass
a = randompoly()
<pre>publickey = balancedmod(3 *</pre>
<pre>convolution(a,dq),q)</pre>
secretkey = d,d3
return publickey,secretkey

sage: A,secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage:

17
<pre>def keypair():</pre>
while True:
try:
<pre>d = randompoly()</pre>
<pre>d3 = invertmodprime(d,3)</pre>
<pre>dq = invertmodpowerof2(d,q)</pre>
break
except:
pass
a = randompoly()
<pre>publickey = balancedmod(3 *</pre>
<pre>convolution(a,dq),q)</pre>
secretkey = d,d3
return publickey,secretkey

sage: A,secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^{6} + x^{5} - x^{4} + x^{3} - 1$ sage:

17
lef keypair():
while True:
try:
<pre>d = randompoly()</pre>
<pre>d3 = invertmodprime(d,3)</pre>
<pre>dq = invertmodpowerof2(d,q)</pre>
break
except:
pass
a = randompoly()
<pre>publickey = balancedmod(3 *</pre>
<pre>convolution(a,dq),q)</pre>
secretkey = d,d3
return publickey,secretkey

sage: A,secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^{6} + x^{5} - x^{4} + x^{3} - 1$ sage: convolution(d,A) $-3 \times x^{6} + 253 \times x^{5} + 253 \times x^{3} 253 \times x^2 - 3 \times x - 3$ sage:

17
<pre>def keypair():</pre>
while True:
try:
<pre>d = randompoly()</pre>
<pre>d3 = invertmodprime(d,3)</pre>
<pre>dq = invertmodpowerof2(d,q)</pre>
break
except:
pass
a = randompoly()
<pre>publickey = balancedmod(3 *</pre>
<pre>convolution(a,dq),q)</pre>
secretkey = d,d3
return publickey,secretkey

sage: A,secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^{6} + x^{5} - x^{4} + x^{3} - 1$ sage: convolution(d,A) $-3 \times x^{6} + 253 \times x^{5} + 253 \times x^{3} -$ 253*x^2 - 3*x - 3 sage: balancedmod(_,q) $-3 \times x^{6} - 3 \times x^{5} - 3 \times x^{3} + 3 \times x^{2}$ - 3*x - 3 sage:

```
17
pair():
                                 sage: A,secretkey = keypair()
                                 sage: A
True:
                                 -126*x^6 - 31*x^5 - 118*x^4 -
•
                                  33*x^3 + 73*x^2 - 16*x + 7
= randompoly()
3 = invertmodprime(d,3)
                                 sage: d,d3 = secretkey
q = invertmodpowerof2(d,q)
                                 sage: d
                                 -x^{6} + x^{5} - x^{4} + x^{3} - 1
reak
                                 sage: convolution(d,A)
ept:
                                 -3 \times x^{6} + 253 \times x^{5} + 253 \times x^{3} -
ass
                                  253*x^2 - 3*x - 3
andompoly()
ckey = balancedmod(3 *
                                 sage: balancedmod(_,q)
                                 -3 \times x^{6} - 3 \times x^{5} - 3 \times x^{3} + 3 \times x^{2}
      convolution(a,dq),q)
                                  - 3*x - 3
tkey = d, d3
n publickey, secretkey
                                 sage:
```

18

NTRU e

One mo w, posit

poly()
<pre>tmodprime(d,3)</pre>
<pre>tmodpowerof2(d,q)</pre>
()
lancedmod(3 *
volution(a,dq),q
d3
ey,secretkey

17

sage: A, secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^{6} + x^{5} - x^{4} + x^{3} - 1$ sage: convolution(d,A) -3*x^6 + 253*x^5 + 253*x^3 -253*x^2 - 3*x - 3 sage: balancedmod(_,q) $-3*x^6 - 3*x^5 - 3*x^3 + 3*x^2$ - 3*x - 3 sage:

NTRU encryption

One more parameter w, positive integer

17	18	
	<pre>sage: A,secretkey = keypair()</pre>	<u>NTRU</u>
	sage: A	One m
	-126*x^6 - 31*x^5 - 118*x^4 -	w, pos
	33*x^3 + 73*x^2 - 16*x + 7	<i>w</i> , pos
(d,3)	<pre>sage: d,d3 = secretkey</pre>	
of2(d,q)	sage: d	
	$-x^{6} + x^{5} - x^{4} + x^{3} - 1$	
	<pre>sage: convolution(d,A)</pre>	
	-3*x^6 + 253*x^5 + 253*x^3 -	
	253*x^2 - 3*x - 3	
.(3 *	<pre>sage: balancedmod(_,q)</pre>	
a,dq),q)	-3*x^6 - 3*x^5 - 3*x^3 + 3*x^2	
	- 3*x - 3	
key	sage:	

l encryption

nore parameter: sitive integer (e.g., 46⁻

sage: A, secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^6 + x^5 - x^4 + x^3 - 1$ sage: convolution(d,A) -3*x^6 + 253*x^5 + 253*x^3 - $253 \times x^2 - 3 \times x - 3$ sage: balancedmod(_,q) $-3 \times x^{6} - 3 \times x^{5} - 3 \times x^{3} + 3 \times x^{2}$ - 3*x - 3

sage:

18

NTRU encryption

One more parameter: w, positive integer (e.g., 467).

sage: A, secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^{6} + x^{5} - x^{4} + x^{3} - 1$ sage: convolution(d,A) -3*x^6 + 253*x^5 + 253*x^3 - $253 \times x^2 - 3 \times x - 3$ sage: balancedmod(_,q) $-3 \times x^{6} - 3 \times x^{5} - 3 \times x^{3} + 3 \times x^{2}$ - 3*x - 3

sage:

NTRU encryption

18

One more parameter: w, positive integer (e.g., 467).

Message for encryption: *n*-coeff weight-*w* polynomial *c* with all coeffs in $\{-1, 0, 1\}$.

"Weight w": w nonzero coeffs, n - w zero coeffs.

sage: A,secretkey = keypair() sage: A -126*x^6 - 31*x^5 - 118*x^4 - $33*x^3 + 73*x^2 - 16*x + 7$ sage: d,d3 = secretkey sage: d $-x^{6} + x^{5} - x^{4} + x^{3} - 1$ sage: convolution(d,A) -3*x^6 + 253*x^5 + 253*x^3 - $253 \times x^2 - 3 \times x - 3$ sage: balancedmod(_,q) $-3 \times x^{6} - 3 \times x^{5} - 3 \times x^{3} + 3 \times x^{2}$ - 3*x - 3 sage:

NTRU encryption

18

One more parameter: w, positive integer (e.g., 467).

Message for encryption: *n*-coeff weight-*w* polynomial *c* with all coeffs in $\{-1, 0, 1\}$.

"Weight w": w nonzero coeffs, n - w zero coeffs.

Ciphertext: C = Ab + c in R_q where b is chosen randomly from the set of messages.

,secretkey = keypair()

6 - 31*x⁵ - 118*x⁴ - $+ 73 \times 2 - 16 \times x + 7$,d3 = secretkey

 $x^5 - x^4 + x^3 - 1$ onvolution(d,A)

+ 253*x^5 + 253*x^3 -

 $2 - 3 \times x - 3$

alancedmod(_,q)

 $- 3 \times x^5 - 3 \times x^3 + 3 \times x^2$

- 3

NTRU encryption

18

One more parameter: w, positive integer (e.g., 4

Message for encryption: *n*-coeff weight-*w* polynomi with all coeffs in $\{-1, 0, 1\}$

"Weight w": w nonzero c n - w zero coeffs.

Ciphertext: C = Ab + c in where *b* is chosen random from the set of messages.

	19		
		sage:	d
		••••	
467).		• • • • •	
101).		• • • • •	
		• • • • •	
nial c		• • • • •	
.}.		• • • • •	
coeffs,		• • • • •	
-		• • • • •	
~		• • • • •	
n R_q		• • • • •	
ly		sage:	W
		sage:	r
		-x^6 -	
		sage:	

y = keypair()

18

5 - 118*x^4 -- 16*x + 7 retkey

+ x^3 - 1 n(d,A) + 253*x^3 -3 d(_,q)

 $3*x^3 + 3*x^2$

NTRU encryption

One more parameter: *w*, positive integer (e.g., 467).

Message for encryption: *n*-coeff weight-*w* polynomial *c* with all coeffs in $\{-1, 0, 1\}$.

"Weight w": w nonzero coeffs, n - w zero coeffs.

Ciphertext: C = Ab + c in R_q where b is chosen randomly from the set of messages.

S	a	g	е	•	de	əf	r	a	n	do	Om
•	•	•	•	•		R		:	r	aı	nd
•	•	•	•	•		а	SS	se	r	t	W
•	•	•	•	•		С	=	:	n	*	[0]
•	•	•	•	•		f	or		j	-	in
•	•	•	•	•			Г	ħ	i	le	Э
•	•	•	•	•					r	=	=
•	•	•	•	•					i	f	n
•	•	•	•	•			C	;[r]	II
•	•	•	•	•		r	et	u	r	n	Ζ
•	•	•	•	•							
S	a	g	е	•	W	=	5				
S	a	g	е	•	ra	an	dc	m	m	e	55
_	X	^	6		- 3	x^	5	+		ХÎ	^4
S	a	g	е	•							

ir()	18
-4- 7	
1	
3 -	

NTRU encryption

One more parameter: w, positive integer (e.g., 467). Message for encryption: *n*-coeff weight-*w* polynomial *c* with all coeffs in $\{-1, 0, 1\}$. "Weight w": w nonzero coeffs, n - w zero coeffs. Ciphertext: C = Ab + c in R_a

where *b* is chosen randomly

from the set of messages.

3*x^2

• • • • • • sage: w = 5sage:

• • • • •

19

sage: def randommessage()

- R = randrange
- \ldots assert w <= n
-: c = n*[0]
-: for j in range(w)
-: while True:
 - r = R(n)
 - if not c[r]:
-: c[r] = 1-2*R(2)
- \ldots : return Zx(c)
- sage: randommessage()
- $-x^{6} x^{5} + x^{4} + x^{3} -$

NTRU encryption

One more parameter: w, positive integer (e.g., 467).

Message for encryption: *n*-coeff weight-*w* polynomial *c* with all coeffs in $\{-1, 0, 1\}$.

"Weight w": w nonzero coeffs, n - w zero coeffs.

Ciphertext: C = Ab + c in R_q where b is chosen randomly from the set of messages.

	sage:	def	rando
	• • • • •	R	= ran
	• • • • •	as	sert
	• • • • •	С	= n*[
	• • • • •	fo	r j i
	• • • • •		while
	• • • • •		r =
	• • • • •		if
	• • • • •		c[r] :
	• • • • •	re	turn 2
	• • • • •		
	sage:	w =	5
	sage:	rand	ommes
	-x^6	- x^5	+ x^
	sage:		
1			

19

mmessage(): drange w <= n [0]n range(w): True: R(n)not c[r]: break = 1 - 2 R(2)Zx(c)

20

sage() 4 + x^3 - x^2

ncryption

- re parameter:
- ive integer (e.g., 467).
- e for encryption: weight-w polynomial c coeffs in $\{-1, 0, 1\}$.
- w": w nonzero coeffs, ero coeffs.
- ext: C = Ab + c in R_a
- is chosen randomly
- e set of messages.

sage: def randommessage(): \ldots R = randrange: assert w <= n: c = n*[0]....: for j in range(w): • while True: r = R(n)• • • • • if not c[r]: break • c[r] = 1-2*R(2)• • • • • \ldots : return Zx(c). sage: w = 5sage: randommessage() $-x^6 - x^5 + x^4 + x^3 - x^2$ sage:

19

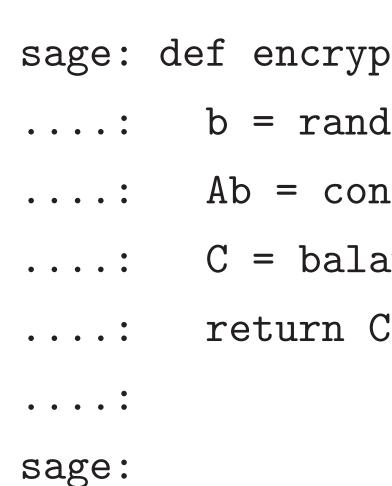
sage: de

- •
- sage:

ter: r (e.g., 467). ption: polynomial c [-1, 0, 1].onzero coeffs, Ab + c in R_a randomly essages.

19

sage: def randommessage(): R = randrange• • • • • assert w <= n • • • • •: c = n * [0]....: for j in range(w): while True: • • • • • r = R(n)• • • • • if not c[r]: break • • • • • c[r] = 1-2*R(2)• • • • •: return Zx(c) • • • • • sage: w = 5sage: randommessage() $-x^{6} - x^{5} + x^{4} + x^{3} - x^{2}$ sage:



7)	•	

19

C

effs,

 R_q

20 sage: def randommessage(): R = randrange • • • • • • • • • • assert w <= n • • • • •: c = n * [0]....: for j in range(w):: while True: • • • • • r = R(n)sage: • • • • • if not c[r]: break • • • • • c[r] = 1-2*R(2)• • • • •: return Zx(c) • • • • sage: w = 5sage: randommessage() $-x^6 - x^5 + x^4 + x^3 - x^2$ sage:

sage: def encrypt(c,A):

- b = randommessage
- \ldots : Ab = convolution(
- \ldots C = balancedmod(A

....: return C

sage:	<pre>def randommessage():</pre>
•	R = randrange
• • • • •	assert w <= n
• • • • •	c = n * [0]
• • • • •	<pre>for j in range(w):</pre>
• • • • •	while True:
• • • • •	r = R(n)
•	if not c[r]: break
• • • • •	c[r] = 1-2*R(2)
•	return Zx(c)
• • • • •	
sage:	w = 5
sage:	<pre>randommessage()</pre>
-x^6 -	- x^5 + x^4 + x^3 - x^2
sage:	

sage: def encrypt(c,A): • • • • • • • • • • • • • return C • • • • • • • • • • sage:

20

21 b = randommessage() Ab = convolution(A,b)C = balancedmod(Ab + c,q)

sage:	<pre>def randommessage():</pre>
•	R = randrange
• • • • •	assert w <= n
• • • • •	c = n * [0]
• • • • •	<pre>for j in range(w):</pre>
• • • • •	while True:
• • • • •	r = R(n)
• • • • •	if not c[r]: break
• • • • •	c[r] = 1-2*R(2)
• • • • •	return Zx(c)
• • • • •	
sage:	w = 5
sage:	<pre>randommessage()</pre>
-x^6 -	$-x^{5} + x^{4} + x^{3} - x^{2}$
sage:	

sage: def encrypt(c,A): • • • • • • • • • • return C • • • • • • • • • • sage: A,secretkey = keypair() sage:

20

21 b = randommessage() Ab = convolution(A,b)C = balancedmod(Ab + c,q)

sage:	<pre>def randommessage():</pre>
• • • • •	R = randrange
• • • • •	assert w <= n
• • • • •	c = n * [0]
• • • • •	<pre>for j in range(w):</pre>
• • • • •	while True:
• • • • •	r = R(n)
• • • • •	if not c[r]: break
•	c[r] = 1-2*R(2)
•	return Zx(c)
•	
sage:	w = 5
sage:	<pre>randommessage()</pre>
-x^6 -	$-x^5 + x^4 + x^3 - x^2$
sage:	

sage:	def encryp
• • • • •	b = rand
• • • • •	Ab = cor
• • • • •	C = bala
• • • • •	return (
• • • • •	
sage:	A,secretke
sage:	c = randon
sage:	

20

21 pt(c,A): dommessage() nvolution(A,b) ancedmod(Ab + c,q) C

ey = keypair() mmessage()

sage:	<pre>def randommessage():</pre>
•	R = randrange
• • • • •	assert w <= n
• • • • •	c = n * [0]
• • • • •	<pre>for j in range(w):</pre>
• • • • •	while True:
• • • • •	r = R(n)
• • • • •	if not c[r]: break
• • • • •	c[r] = 1-2*R(2)
• • • • •	return Zx(c)
• • • • •	
sage:	w = 5
sage:	<pre>randommessage()</pre>
-x^6 -	- x^5 + x^4 + x^3 - x^2
sage:	

sage: def encrypt(c,A): • • • • • \ldots : Ab = convolution(A,b): return C • • • • • sage: A,secretkey = keypair() sage: c = randommessage() sage: C = encrypt(c,A) sage:

20

21 b = randommessage() ...: C = balancedmod(Ab + c,q)

sage:	<pre>def randommessage():</pre>
• • • • •	R = randrange
• • • • •	assert w <= n
•	c = n * [0]
•	<pre>for j in range(w):</pre>
•	while True:
• • • • •	r = R(n)
• • • • •	if not c[r]: break
• • • • •	c[r] = 1-2*R(2)
• • • • •	return Zx(c)
•	
sage:	w = 5
sage:	<pre>randommessage()</pre>
-x^6 -	$-x^{5} + x^{4} + x^{3} - x^{2}$
sage:	

sage:	def	encry
• • • • •	b	= rand
• • • • •	Al	o = cor
• • • • •	С	= bala
• • • • •	re	eturn (
•		
sage:	A,se	ecretke
sage:	с =	randor
sage:	C =	encry
sage:	С	
21*x^6	6 - 4	48*x^5
76*x ⁻	^3 -	77*x^2
sage:		

20

21 pt(c,A): dommessage() nvolution(A,b) ancedmod(Ab + c,q) C

- ey = keypair()
 mmessage()
 pt(c,A)
- + 31*x^4 -2 + 15*x - 113

ef randommessage():
R = randrange
assert w <= n
c = n * [0]
<pre>for j in range(w):</pre>
while True:
r = R(n)
if not c[r]: break
c[r] = 1-2*R(2)
return Zx(c)

20

= 5

andommessage()

 $x^5 + x^4 + x^3 - x^2$

sage: def encrypt(c,A):: b = randommessage() \ldots : Ab = convolution(A,b): C = balancedmod(Ab + c,q)....: return C • • • • • sage: A, secretkey = keypair() sage: c = randommessage() sage: C = encrypt(c,A) sage: C 21*x^6 - 48*x^5 + 31*x^4 - $76*x^3 - 77*x^2 + 15*x - 113$ sage:

21

NTRU c

Compute

20	21
message():	<pre>sage: def encrypt(c,A):</pre>
range	: b = randommessage()
<= n	: $Ab = convolution(A,b)$
]	: $C = balancedmod(Ab + c,q)$
<pre>range(w):</pre>	: return C
True:	• • • •
R(n)	<pre>sage: A,secretkey = keypair()</pre>
ot c[r]: break	<pre>sage: c = randommessage()</pre>
1-2*R(2)	<pre>sage: C = encrypt(c,A)</pre>
x(c)	sage: C
	21*x^6 - 48*x^5 + 31*x^4 -
	76*x^3 - 77*x^2 + 15*x - 113
age()	sage:
+ x^3 - x^2	

NTRU decryption

Compute dC = 3a

20	21	
	<pre>sage: def encrypt(c,A):</pre>	NTRU
	<pre>: b = randommessage()</pre>	Comp
	: $Ab = convolution(A,b)$	comp
	: $C = balancedmod(Ab + c,q)$	
	: return C	
	<pre>sage: A,secretkey = keypair()</pre>	
	<pre>sage: c = randommessage()</pre>	
	<pre>sage: C = encrypt(c,A)</pre>	
	sage: C	
	21*x^6 - 48*x^5 + 31*x^4 -	
	76*x^3 - 77*x^2 + 15*x - 113	
	sage:	

break

•

•

x^2

J decryption

pute dC = 3ab + dc in

sage: def encrypt(c,A):: b = randommessage() \ldots : Ab = convolution(A,b) ...: C = balancedmod(Ab + c,q)....: return C • • • • • sage: A, secretkey = keypair() sage: c = randommessage() sage: C = encrypt(c, A)sage: C 21*x^6 - 48*x^5 + 31*x^4 - $76*x^3 - 77*x^2 + 15*x - 113$ sage:

NTRU decryption

21

Compute dC = 3ab + dc in R_q .

21
<pre>sage: def encrypt(c,A):</pre>
<pre>: b = randommessage()</pre>
: $Ab = convolution(A,b)$
: $C = balancedmod(Ab + c,q)$
: return C
<pre>sage: A,secretkey = keypair()</pre>
<pre>sage: c = randommessage()</pre>
<pre>sage: C = encrypt(c,A)</pre>
sage: C
21*x^6 - 48*x^5 + 31*x^4 -
76*x^3 - 77*x^2 + 15*x - 113
sage:

NTRU decryption

a, b, c, d have small coeffs, so 3ab + dc is not very big.

Compute dC = 3ab + dc in R_q .

21
<pre>sage: def encrypt(c,A):</pre>
<pre>: b = randommessage()</pre>
: $Ab = convolution(A,b)$
: $C = balancedmod(Ab + c,q)$
: return C
• • • •
<pre>sage: A,secretkey = keypair()</pre>
<pre>sage: c = randommessage()</pre>
<pre>sage: C = encrypt(c,A)</pre>
sage: C
21*x^6 - 48*x^5 + 31*x^4 -
76*x^3 - 77*x^2 + 15*x - 113
sage:

NTRU decryption

Compute dC = 3ab + dc in R_q .

a, b, c, d have small coeffs, so 3ab + dc is not very big.

Assume that coeffs of 3ab + dcare between -q/2 and q/2 - 1.

21
<pre>sage: def encrypt(c,A):</pre>
<pre>: b = randommessage()</pre>
: $Ab = convolution(A,b)$
: $C = balancedmod(Ab + c,q)$
: return C
• • • • •
<pre>sage: A,secretkey = keypair()</pre>
<pre>sage: c = randommessage()</pre>
<pre>sage: C = encrypt(c,A)</pre>
sage: C
21*x^6 - 48*x^5 + 31*x^4 -
76*x^3 - 77*x^2 + 15*x - 113
sage:

NTRU decryption Compute dC = 3ab + dc in R_q . a, b, c, d have small coeffs, so 3ab + dc is not very big. **Assume** that coeffs of 3ab + dcare between -q/2 and q/2 - 1. Then 3ab + dc in R_q reveals

3ab + dc in $R = \mathbf{Z}[x]/(x^n - 1)$.

21
<pre>sage: def encrypt(c,A):</pre>
<pre>: b = randommessage()</pre>
\ldots : Ab = convolution(A,b)
: $C = balancedmod(Ab + c,q)$
: return C
• • • •
<pre>sage: A,secretkey = keypair()</pre>
<pre>sage: c = randommessage()</pre>
<pre>sage: C = encrypt(c,A)</pre>
sage: C
21*x^6 - 48*x^5 + 31*x^4 -
76*x^3 - 77*x^2 + 15*x - 113
sage:

NTRU decryption Compute dC = 3ab + dc in R_q . a, b, c, d have small coeffs, so 3ab + dc is not very big. **Assume** that coeffs of 3ab + dcare between -q/2 and q/2 - 1. Then 3ab + dc in R_q reveals 3ab + dc in $R = \mathbf{Z}[x]/(x^n - 1)$. Reduce modulo 3: dc in R_3 .

21
<pre>sage: def encrypt(c,A):</pre>
<pre>: b = randommessage()</pre>
\ldots : Ab = convolution(A,b)
: $C = balancedmod(Ab + c,q)$
: return C
• • • • •
<pre>sage: A,secretkey = keypair()</pre>
<pre>sage: c = randommessage()</pre>
<pre>sage: C = encrypt(c,A)</pre>
sage: C
21*x^6 - 48*x^5 + 31*x^4 -
76*x^3 - 77*x^2 + 15*x - 113
sage:

NTRU decryption Compute dC = 3ab + dc in R_q . a, b, c, d have small coeffs, so 3ab + dc is not very big. **Assume** that coeffs of 3ab + dcare between -q/2 and q/2 - 1. Then 3ab + dc in R_q reveals 3ab + dc in $R = Z[x]/(x^n - 1)$. Reduce modulo 3: dc in R_3 . Multiply by 1/d in R_3 to recover message c in R_3 .

21
<pre>sage: def encrypt(c,A):</pre>
<pre>: b = randommessage()</pre>
\ldots : Ab = convolution(A,b)
: $C = balancedmod(Ab + c,q)$
: return C
<pre>sage: A,secretkey = keypair()</pre>
<pre>sage: c = randommessage()</pre>
<pre>sage: C = encrypt(c,A)</pre>
sage: C
21*x^6 - 48*x^5 + 31*x^4 -
76*x^3 - 77*x^2 + 15*x - 113
sage:

NTRU decryption Compute dC = 3ab + dc in R_q . a, b, c, d have small coeffs, so 3ab + dc is not very big. **Assume** that coeffs of 3ab + dcare between -q/2 and q/2 - 1. Then 3ab + dc in R_q reveals 3ab + dc in $R = Z[x]/(x^n - 1)$. Reduce modulo 3: dc in R_3 . Multiply by 1/d in R_3 to recover message c in R_3 . Coeffs are between -1 and 1, so recover c in R.

ef encrypt(c,A):

- b = randommessage()
- Ab = convolution(A,b)
- C = balancedmod(Ab + c,q)
- return C

,secretkey = keypair()

- = randommessage()
- = encrypt(c,A)
- 48*x^5 + 31*x^4 -
- 77*x^2 + 15*x 113

NTRU decryption

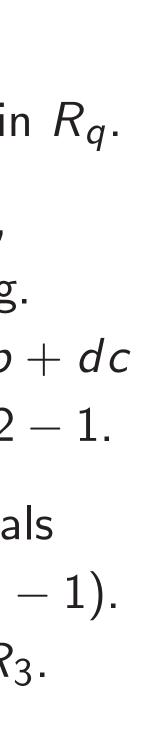
21

Compute dC = 3ab + dc in R_q .

a, b, c, d have small coeffs, so 3ab + dc is not very big. Assume that coeffs of 3ab + dcare between -q/2 and q/2 - 1.

Then 3ab + dc in R_q reveals 3ab + dc in $R = \mathbf{Z}[x]/(x^n - 1)$. Reduce modulo 3: dc in R_3 .

Multiply by 1/d in R_3 to recover message c in R_3 . Coeffs are between -1 and 1, so recover c in R.



sage: d

- sage:

```
t(c,A):
ommessage()
volution(A,b)
ncedmod(Ab + c,q)
y = keypair()
message()
t(c,A)
```

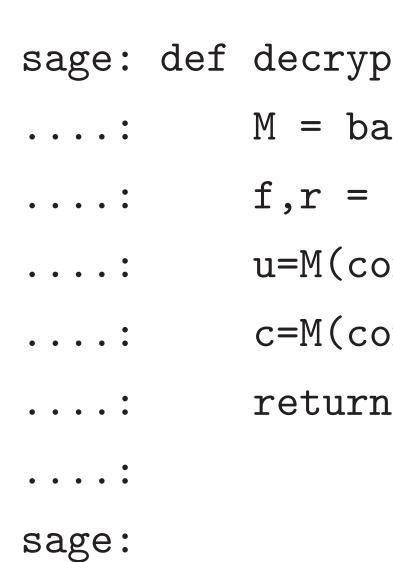
+ 31*x^4 -

+ 15*x - 113

NTRU decryption

21

Compute dC = 3ab + dc in R_q . a, b, c, d have small coeffs, so 3ab + dc is not very big. **Assume** that coeffs of 3ab + dcare between -q/2 and q/2 - 1. Then 3ab + dc in R_q reveals 3ab + dc in $R = Z[x]/(x^n - 1)$. Reduce modulo 3: dc in R_3 . Multiply by 1/d in R_3 to recover message c in R_3 . Coeffs are between -1 and 1, so recover c in R.



21	
	<u>N</u> TF
()	Com
A,b)	COII
b + c,q)	a, b,
	so 3
	Assi
ir()	are l
	The
	3ab
	Red
- 113	Mult
	to re
	Coe
	so re

RU decryption npute dC = 3ab + dc in R_a . c, d have small coeffs, bab + dc is not very big. ume that coeffs of 3ab + dcbetween -q/2 and q/2 - 1. n 3ab + dc in R_q reveals $+ dc \text{ in } R = \mathbf{Z}[x]/(x^{n} - 1).$ uce modulo 3: dc in R_3 . tiply by 1/d in R_3 ecover message c in R_3 . ffs are between -1 and 1, ecover *c* in *R*.

. • sage:

22

- sage: def decrypt(C,secre
 - M = balancedmod
 - f,r = secretkey
 - u=M(convolution
 - c=M(convolution

return c

NTRU decryption

Compute dC = 3ab + dc in R_a .

a, b, c, d have small coeffs, so 3ab + dc is not very big. **Assume** that coeffs of 3ab + dcare between -q/2 and q/2 - 1.

Then
$$3ab + dc$$
 in R_q reveals
 $3ab + dc$ in $R = \mathbf{Z}[x]/(x^n - 1)$.
Reduce modulo 3: dc in R_3 .

Multiply by 1/d in R_3 to recover message c in R_3 . Coeffs are between -1 and 1, so recover c in R.

• return c • • • • •

sage:

22

sage: def decrypt(C,secretkey):

M = balancedmod

f,r = secretkey

u=M(convolution(C,f),q) c=M(convolution(u,r),3)

NTRU decryption

Compute dC = 3ab + dc in R_q .

a, b, c, d have small coeffs, so 3ab + dc is not very big. Assume that coeffs of 3ab + dcare between -q/2 and q/2 - 1.

Then
$$3ab + dc$$
 in R_q reveals
 $3ab + dc$ in $R = \mathbf{Z}[x]/(x^n - 1)$.
Reduce modulo 3: dc in R_3 .

Multiply by 1/d in R_3 to recover message c in R_3 . Coeffs are between -1 and 1, so recover c in R.

sage:	def	decry
• • • • •		M = b
• • • • •		f,r =
• • • • •		u=M(c
• • • • •		c=M(c
• • • • •		retur
• • • • •		
sage:	С	
x^5 +	x^4	- x^3
sage:		

22

rpt(C,secretkey):

alancedmod

secretkey

onvolution(C,f),q)
onvolution(u,r),3)

n c

+ x + 1

NTRU decryption

Compute dC = 3ab + dc in R_q .

a, b, c, d have small coeffs, so 3ab + dc is not very big. Assume that coeffs of 3ab + dcare between -q/2 and q/2 - 1.

Then
$$3ab + dc$$
 in R_q reveals
 $3ab + dc$ in $R = \mathbf{Z}[x]/(x^n - 1)$.
Reduce modulo 3: dc in R_3 .

Multiply by 1/d in R_3 to recover message c in R_3 . Coeffs are between -1 and 1, so recover c in R.

sage:	def	decry
• • • • •		M = b
• • • • •		f,r =
• • • • •		u=M(c
• • • • •		c=M(c
• • • • •		retur
• • • • •		
sage:	С	
x^5 +	x^4	- x^3
sage:	decı	cypt(C
x^5 +	x^4	- x^3
sage:		

22

rpt(C,secretkey):

23

alancedmod

secretkey

onvolution(C,f),q)
onvolution(u,r),3)

n c

+ x + 1

,secretkey)

+ x + 1

lecryption

- $e dC = 3ab + dc in R_q$.
- have small coeffs, - *dc* is not very big. that coeffs of 3ab + dcveen -q/2 and q/2-1.
- b + dc in R_a reveals c in $R = \mathbf{Z}[x]/(x^n - 1)$. modulo 3: dc in R_3 .
- by 1/d in R_3 er message c in R_3 . re between -1 and 1, er c in R.

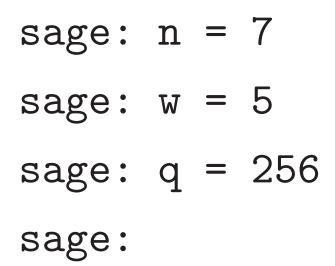
sage: def decrypt(C,secr M = balancedmo• • • • f,r = secretke • • • • • u=M(convolutio • • • • • c=M(convolutio • • return c • • • • • sage: c $x^5 + x^4 - x^3 + x + 1$ sage: decrypt(C,secretkey) $x^5 + x^4 - x^3 + x + 1$ sage:

	23
retkey):	
d	
ey y	
on(C,f),	q)
on(u,r),3	3)

- sage: n sage: w
- sage: q
- sage:

	22		
		sage:	def
$b + dc$ in R_q .		• • • • •	
9		• • • • •	
Il coeffs,		• • • • •	
t very big.		•	
fs of $3ab + dc$		• • • • •	
and $q/2 - 1$.		•	
R_q reveals		sage:	С
$Z[x]/(x^n-1).$		x^5 +	x^4
dc in R_3 .		sage:	dec
		x^5 +	x^4
R_3		sage:	
$e c in R_3.$			
n-1 and 1,			
4			

		23
sage:	def	<pre>decrypt(C,secretkey):</pre>
• • • • •		M = balancedmod
• • • • •		f,r = secretkey
• • • • •		u=M(convolution(C,f),q)
• • • • •		c=M(convolution(u,r),3)
• • • • •		return c
• • • • •		
sage:	С	
x^5 +	x^4	$-x^3 + x + 1$
sage:	decr	<pre>ypt(C,secretkey)</pre>
x^5 +	x^4	$-x^{3} + x + 1$
sage:		



22	23		
	<pre>sage: def decrypt(C,secretkey):</pre>	sage:	n
R_q .	\ldots : M = balancedmod	sage:	W
<i>r</i> q ·	: f,r = secretkey	sage:	q
	<pre>: u=M(convolution(C,f),q)</pre>	sage:	
	: c=M(convolution(u,r),3)		
+ dc	: return c		
- 1.	• • • •		
S	sage: c		
- 1).	$x^5 + x^4 - x^3 + x + 1$		
-	<pre>sage: decrypt(C,secretkey)</pre>		
	$x^5 + x^4 - x^3 + x + 1$		
	sage:		
-			
1,			

n = 7w = 5q = 256

		۷. ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰۰۰ ۲۰
sage:	def	<pre>decrypt(C,secretkey):</pre>
• • • • •		M = balancedmod
• • • • •		f,r = secretkey
• • • • •		u=M(convolution(C,f),q
• • • • •		c=M(convolution(u,r),3
• • • • •		return c
• • • • •		
sage:	С	
x^5 +	x^4	$-x^3 + x + 1$
sage:	deci	rypt(C,secretkey)
x^5 +	x^4	$-x^3 + x + 1$
sage:		

sage: n = 7
sage: w = 5
sage: q = 256
sage:

23

)

		20
sage:	def	<pre>decrypt(C,secretkey):</pre>
• • • • •		M = balancedmod
• • • • •		f,r = secretkey
• • • • •		u=M(convolution(C,f),q)
• • • • •		c=M(convolution(u,r),3)
• • • • •		return c
• • • • •		
sage:	С	
x^5 +	x^4	$-x^3 + x + 1$
sage:	deci	rypt(C,secretkey)
x^5 +	x^4	$-x^3 + x + 1$
sage:		

sage: n = 7sage: w = 5sage: q = 256sage: A,secretkey = keypair() sage:

23

23
<pre>sage: def decrypt(C,secretkey):</pre>
: M = balancedmod
: f,r = secretkey
<pre>: u=M(convolution(C,f),q)</pre>
: c=M(convolution(u,r),3)
: return c
sage: c
$x^5 + x^4 - x^3 + x + 1$
<pre>sage: decrypt(C,secretkey)</pre>
$x^5 + x^4 - x^3 + x + 1$
sage:

```
sage: n = 7
sage: w = 5
sage: q = 256
sage: A,secretkey = keypair()
sage: A
-101*x^6 - 76*x^5 - 90*x^4 -
83*x^3 + 40*x^2 + 108*x - 54
sage:
```

23
<pre>sage: def decrypt(C,secretkey):</pre>
: M = balancedmod
: f,r = secretkey
: u=M(convolution(C,f),q)
: c=M(convolution(u,r),3)
: return c
• • • •
sage: c
$x^5 + x^4 - x^3 + x + 1$
<pre>sage: decrypt(C,secretkey)</pre>
$x^5 + x^4 - x^3 + x + 1$
sage:

sage: n = 7sage: w = 5sage: q = 256sage: A,secretkey = keypair() sage: A $-101*x^6 - 76*x^5 - 90*x^4 83*x^3 + 40*x^2 + 108*x - 54$ sage: d,d3 = secretkey sage:

	23	
sage: def	<pre>decrypt(C,secretkey):</pre>	sage: $n = 7$
• • • • •	M = balancedmod	sage: $w = 5$
• • • • •	f,r = secretkey	sage: q = 256
• • • • •	u=M(convolution(C,f),q)	sage: A,secret
• • • • •	<pre>c=M(convolution(u,r),3)</pre>	sage: A
• • • • •	return c	-101*x^6 - 76*
• • • • •		83*x^3 + 40*x
sage: c		sage: d,d3 = s
x^5 + x^4	$-x^3 + x + 1$	sage: d
sage: dec	rypt(C,secretkey)	x^5 + x^4 - x^
x^5 + x^4	$-x^3 + x + 1$	sage:
sage:		

retkey = keypair()

- 76*x^5 90*x^4 -
- $0*x^2 + 108*x 54$
- = secretkey

$x^3 + x - 1$

	23	
sage: de	<pre>f decrypt(C,secretkey):</pre>	sage: $n = 7$
• • • • •	M = balancedmod	sage: $w = 5$
• • • • •	f,r = secretkey	sage: q = 256
• • • • •	u=M(convolution(C,f),q)	sage: A, secret
• • • • •	<pre>c=M(convolution(u,r),3)</pre>	sage: A
• • • • •	return c	-101*x^6 - 76*:
• • • • •		83*x^3 + 40*x
sage: c		sage: $d,d3 = se$
x^5 + x^	$4 - x^3 + x + 1$	sage: d
sage: de	crypt(C,secretkey)	$x^5 + x^4 - x^3$
x^5 + x^	$4 - x^3 + x + 1$	sage: conv = co
sage:		sage:

tkey = keypair()

- *x^5 90*x^4 -
- $x^2 + 108 * x 54$
- secretkey

3 + x - 1convolution

	23	
sage: def	<pre>decrypt(C,secretkey):</pre>	sage: $n = 7$
• • • • •	M = balancedmod	sage: $w = 5$
• • • • •	f,r = secretkey	sage: $q = 256$
• • • • •	u=M(convolution(C,f),q)	sage: A,secret
• • • • •	c=M(convolution(u,r),3)	sage: A
• • • • •	return c	-101*x^6 - 76*
•		83*x^3 + 40*x
sage: c		sage: $d,d3 = s$
x^5 + x^4	$-x^3 + x + 1$	sage: d
sage: decr	<pre>cypt(C,secretkey)</pre>	x^5 + x^4 - x^
x^5 + x^4	$-x^3 + x + 1$	<pre>sage: conv = c</pre>
sage:		sage: M = bala

retkey = keypair()

- 76*x^5 90*x^4 -
- $3 \times 2 + 108 \times 2 54$
- = secretkey

$x^3 + x - 1$ = convolution alancedmod

sage:

23	
<pre>sage: def decrypt(C,secretkey):</pre>	sage: n =
: M = balancedmod	sage: w =
: f,r = secretkey	sage: q =
<pre>: u=M(convolution(C,f),q)</pre>	sage: A,se
: c=M(convolution(u,r),3)	sage: A
: return c	-101*x^6 -
• • • •	83*x^3 +
sage: c	sage: d,d3
$x^5 + x^4 - x^3 + x + 1$	sage: d
<pre>sage: decrypt(C,secretkey)</pre>	x^5 + x^4
$x^5 + x^4 - x^3 + x + 1$	sage: conv
sage:	<pre>sage: M =</pre>
	_

ecretkey = keypair()

- 76*x^5 90*x^4 -
 - 40*x^2 + 108*x 54
- 13 = secretkey

7

5

sage:

- x^3 + x 1
- v = convolution
 - balancedmod
- sage: a3 = M(conv(d,A),q)

23	
<pre>sage: def decrypt(C,secretkey):</pre>	sage: n =
\ldots : M = balancedmod	<pre>sage: w =</pre>
: f,r = secretkey	sage: q =
<pre>: u=M(convolution(C,f),q)</pre>	sage: A,se
: c=M(convolution(u,r),3)	sage: A
: return c	-101*x^6 -
• • • •	83*x^3 +
sage: c	sage: d,d3
$x^5 + x^4 - x^3 + x + 1$	sage: d
<pre>sage: decrypt(C,secretkey)</pre>	x^5 + x^4
$x^5 + x^4 - x^3 + x + 1$	sage: conv
sage:	sage: $M =$
	sage: a3 =

ecretkey = keypair()

- 76*x^5 90*x^4 -
 - $40*x^2 + 108*x 54$
- 3 = secretkey

7

5

sage: a3

 $3*x^2 - 3*x$

- x^3 + x 1
- v = convolution
 - balancedmod
- sage: a3 = M(conv(d,A),q)

	23
ef	<pre>decrypt(C,secretkey):</pre>
	M = balancedmod
	f,r = secretkey
	u=M(convolution(C,f),q)
	c=M(convolution(u,r),3)
	return c
^4	$-x^3 + x + 1$
ecı	<pre>cypt(C,secretkey)</pre>
^4	$-x^3 + x + 1$

	24		
		sage:	С
		sage:	
pair()			
c^4 -			
z – 54			
L			
L)			

23	
t(C,secretkey):	sage: $n = 7$
lancedmod	sage: $w = 5$
secretkey	sage: q = 256
<pre>nvolution(C,f),q)</pre>	<pre>sage: A,secretkey = keypair()</pre>
nvolution(u,r),3)	sage: A
С	-101*x^6 - 76*x^5 - 90*x^4 -
	83*x^3 + 40*x^2 + 108*x - 54
	<pre>sage: d,d3 = secretkey</pre>
+ x + 1	sage: d
secretkey)	$x^5 + x^4 - x^3 + x - 1$
+ x + 1	<pre>sage: conv = convolution</pre>
	<pre>sage: M = balancedmod</pre>
	sage: $a3 = M(conv(d,A),q)$
	sage: a3
	3*x^2 - 3*x

sage: c = random

sage:

23		24		
tkey):	sage: $n = 7$		sage:	C
	sage: $w = 5$		sage:	
	sage: q = 256			
(C,f),q)	<pre>sage: A,secretkey = keypair()</pre>			
(u,r),3)	sage: A			
	-101*x^6 - 76*x^5 - 90*x^4 -			
	83*x^3 + 40*x^2 + 108*x - 54			
	<pre>sage: d,d3 = secretkey</pre>			
	sage: d			
•)	$x^5 + x^4 - x^3 + x - 1$			
	<pre>sage: conv = convolution</pre>			
	<pre>sage: M = balancedmod</pre>			
	<pre>sage: a3 = M(conv(d,A),q)</pre>			
	sage: a3			
	3*x^2 - 3*x			
		1		



sage: c = randommessage() sage:

sage: c = randommessage() sage: b = randommessage() sage:

24

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage:

24

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage: C $-57*x^{6} + 28*x^{5} + 114*x^{4} +$ $72*x^3 - 37*x^2 + 16*x + 119$ sage:

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage: C $-57*x^{6} + 28*x^{5} + 114*x^{4} +$ $72*x^3 - 37*x^2 + 16*x + 119$ sage: u = M(conv(C,d),q)sage:

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage: C $-57*x^{6} + 28*x^{5} + 114*x^{4} +$ $72*x^3 - 37*x^2 + 16*x + 119$ sage: u = M(conv(C,d),q)sage: u $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - 3$ $4*x^2 + 5*x + 1$ sage:

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage: C $-57*x^{6} + 28*x^{5} + 114*x^{4} +$ $72*x^3 - 37*x^2 + 16*x + 119$ sage: u = M(conv(C,d),q)sage: u $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3$ $4*x^2 + 5*x + 1$ sage: conv(a3,b)+conv(c,d) $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3$ $4*x^2 + 5*x + 1$

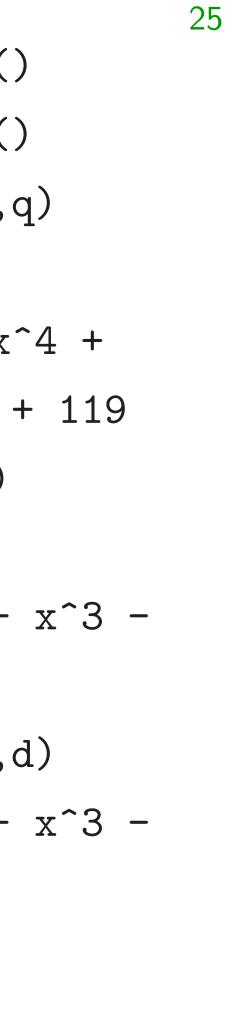
24

= 7
= 5
= 256
<pre>,secretkey = keypair()</pre>
6 - 76*x^5 - 90*x^4 -
+ 40*x^2 + 108*x - 54
,d3 = secretkey
^4 - x^3 + x - 1
onv = convolution
= balancedmod
B = M(conv(d,A),q)
3

3*x

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage: C $-57*x^{6} + 28*x^{5} + 114*x^{4} +$ $72*x^3 - 37*x^2 + 16*x + 119$ sage: u = M(conv(C,d),q)sage: u $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3$ $4*x^2 + 5*x + 1$ sage: conv(a3,b)+conv(c,d) $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3$ $4*x^2 + 5*x + 1$

24



sage: M $x^6 - x$ + 1 sage:

	24	
		sage:
		sage:
		sage:
y = keypair()		sage:
		-57*x*
5 - 90*x^4 -		72*x [*]
+ 108*x - 54		sage:
retkey		sage:
		-8*x^(
+ x - 1		4*x^2
volution		sage:
edmod		-8*x^(
v(d,A),q)		4*x^2

<pre>sage: c = randommessage()</pre>	
<pre>sage: b = randommessage()</pre>	
sage: $C = M(conv(A,b)+c,q)$	
sage: C	
-57*x^6 + 28*x^5 + 114*x^4 +	
72*x^3 - 37*x^2 + 16*x + 119	
<pre>sage: u = M(conv(C,d),q)</pre>	
sage: u	
-8*x^6 + 2*x^5 + 4*x^4 - x^3 -	
$4*x^2 + 5*x + 1$	
<pre>sage: conv(a3,b)+conv(c,d)</pre>	
-8*x^6 + 2*x^5 + 4*x^4 - x^3 -	
$4*x^2 + 5*x + 1$	

sage: M(u,3) x^6 - x^5 + x^4 + 1 sage:

	25
<pre>sage: c = randommessage()</pre>	
<pre>sage: b = randommessage()</pre>	
sage: $C = M(conv(A,b)+c,q)$	
sage: C	
-57*x^6 + 28*x^5 + 114*x^4 +	
72*x^3 - 37*x^2 + 16*x + 119	
<pre>sage: u = M(conv(C,d),q)</pre>	
sage: u	
-8*x^6 + 2*x^5 + 4*x^4 - x^3 -	
4*x^2 + 5*x + 1	
<pre>sage: conv(a3,b)+conv(c,d)</pre>	
-8*x^6 + 2*x^5 + 4*x^4 - x^3 -	
4*x^2 + 5*x + 1	

sage: M(u,3) + 1

sage:

ir()

24

4 -

- 54

 $x^6 - x^5 + x^4 - x^3 - x$

```
sage: c = randommessage()
sage: b = randommessage()
sage: C = M(conv(A,b)+c,q)
sage: C
-57*x^{6} + 28*x^{5} + 114*x^{4} +
 72*x^3 - 37*x^2 + 16*x + 119
sage: u = M(conv(C,d),q)
sage: u
-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - 3
 4*x^2 + 5*x + 1
sage: conv(a3,b)+conv(c,d)
-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3
 4*x^2 + 5*x + 1
```

```
sage: M(u,3)
x^6 - x^5 + x^4 - x^3 - x^2 - x
 + 1
sage:
```

```
sage: c = randommessage()
sage: b = randommessage()
sage: C = M(conv(A,b)+c,q)
sage: C
-57*x^{6} + 28*x^{5} + 114*x^{4} +
 72*x^3 - 37*x^2 + 16*x + 119
sage: u = M(conv(C,d),q)
sage: u
-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - 3
 4*x^2 + 5*x + 1
sage: conv(a3,b)+conv(c,d)
-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3
 4*x^2 + 5*x + 1
```

```
sage: M(u,3)
 + 1
sage: M(conv(c,d),3)
+ 1
sage:
```

$x^6 - x^5 + x^4 - x^3 - x^2 - x$ $x^6 - x^5 + x^4 - x^3 - x^2 - x$

sage: c = randommessage()
sage: b = randommessage()
sage: C = M(conv(A,b)+c,q)
sage: C
$$-57*x^{6} + 28*x^{5} + 114*x^{4} + 72*x^{3} - 37*x^{2} + 16*x + 119
sage: u = M(conv(C,d),q)
sage: u
 $-8*x^{6} + 2*x^{5} + 4*x^{4} - x^{3}$
 $4*x^{2} + 5*x + 1$
sage: conv(a3,b)+conv(c,d)
 $-8*x^{6} + 2*x^{5} + 4*x^{4} - x^{3}$
 $4*x^{2} + 5*x + 1$$$

sage: M(u,3)+ 1 sage: M(conv(c,d),3) + 1 sage: conv(M(u,3),d3) $x^6 - x^5 - x^4 - 3 x^3 - x^2 +$ x - 3 sage:

25

26

$x^6 - x^5 + x^4 - x^3 - x^2 - x$ $x^6 - x^5 + x^4 - x^3 - x^2 - x$

sage: c = randommessage()
sage: b = randommessage()
sage: C = M(conv(A,b)+c,q)
sage: C
$$-57*x^{6} + 28*x^{5} + 114*x^{4} + 72*x^{3} - 37*x^{2} + 16*x + 119
sage: u = M(conv(C,d),q)
sage: u
 $-8*x^{6} + 2*x^{5} + 4*x^{4} - x^{3} + 4*x^{2} + 5*x + 1
sage: conv(a3,b)+conv(c,d)
 $-8*x^{6} + 2*x^{5} + 4*x^{4} - x^{3} + 4*x^{2} + 5*x + 1$$$$

sage: M(u,3) $x^6 - x^5 + x^4 - x^3 - x^2 - x$ + 1 sage: M(conv(c,d),3) + 1 sage: conv(M(u,3),d3)x - 3 sage: M(_,3) $x^6 - x^5 - x^4 - x^2 + x$ sage:

25

26

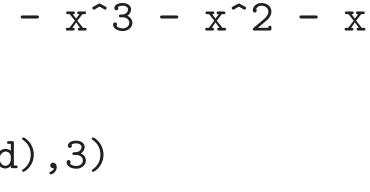
 $x^6 - x^5 + x^4 - x^3 - x^2 - x$

$x^6 - x^5 - x^4 - 3 x^3 - x^2 +$

sage: c = randommessage() sage: b = randommessage() sage: C = M(conv(A,b)+c,q)sage: C $-57*x^{6} + 28*x^{5} + 114*x^{4} +$ $72*x^3 - 37*x^2 + 16*x + 119$ sage: u = M(conv(C,d),q)sage: u $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3$ $4*x^2 + 5*x + 1$ sage: conv(a3,b)+conv(c,d) $-8 \times 6 + 2 \times 5 + 4 \times 4 - x^3 - x^3$ $4*x^2 + 5*x + 1$

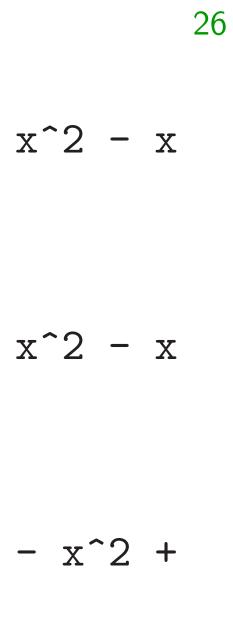
25

sage: M(u,3) $x^6 - x^5 + x^4 - x^3 - x^2 - x$ + 1 sage: M(conv(c,d),3) $x^6 - x^5 + x^4 - x^3 - x^2 - x$ + 1 sage: conv(M(u,3),d3) $x^6 - x^5 - x^4 - 3 x^3 - x^2 +$ x - 3 sage: $M(_,3)$ $x^6 - x^5 - x^4 - x^2 + x$ sage: c $x^6 - x^5 - x^4 - x^2 + x$ sage:



=	<pre>randommessage()</pre>
=	randommessage()
=	M(conv(A,b)+c,q)
+	28*x^5 + 114*x^4 +
_	37*x^2 + 16*x + 119
=	M(conv(C,d),q)
+ 2	2*x^5 + 4*x^4 - x^3 -
+ 5	5*x + 1
onv	/(a3,b)+conv(c,d)
+ 2	2*x^5 + 4*x^4 - x^3 -
+ 5	5*x + 1

sage: M(u,3) $x^6 - x^5 + x^4 - x^3 - x^2 - x$ + 1 sage: M(conv(c,d),3) $x^6 - x^5 + x^4 - x^3 - x^2 - x$ + 1 sage: conv(M(u,3),d3) $x^6 - x^5 - x^4 - 3 x^3 - x^2 +$ x - 3 sage: M(_,3) $x^6 - x^5 - x^4 - x^2 + x$ sage: c $x^6 - x^5 - x^4 - x^2 + x$ sage:



Does de

All coeff All coeff and exac

	25		26
message()		sage: M(u,3)	
message()		$x^6 - x^5 + x^4 - x^3 - x^2 - x$	
(A,b)+c,q)		+ 1	
		<pre>sage: M(conv(c,d),3)</pre>	
+ 114*x^4 +		$x^6 - x^5 + x^4 - x^3 - x^2 - x$	-
+ 16*x + 119		+ 1	
(C,d),q)		<pre>sage: conv(M(u,3),d3)</pre>	
		x^6 - x^5 - x^4 - 3*x^3 - x^2 +	-
4*x^4 - x^3 -		x - 3	
		sage: M(_,3)	
+conv(c,d)		$x^6 - x^5 - x^4 - x^2 + x$	
4*x^4 - x^3 -		sage: c	
		$x^6 - x^5 - x^4 - x^2 + x$	
		sage:	

Does decryption a

All coeffs of a are All coeffs of b are and exactly w are

	25	26	
		sage: M(u,3)	Does c
		$x^6 - x^5 + x^4 - x^3 - x^2 - x$	All coe
)		+ 1	All coe
		<pre>sage: M(conv(c,d),3)</pre>	and ex
4 +		$x^6 - x^5 + x^4 - x^3 - x^2 - x$	
119		+ 1	
		<pre>sage: conv(M(u,3),d3)</pre>	
		x^6 - x^5 - x^4 - 3*x^3 - x^2 +	
x^3 -		x - 3	
		sage: M(_,3)	
)		$x^6 - x^5 - x^4 - x^2 + x$	
x^3 -		sage: c	
		$x^6 - x^5 - x^4 - x^2 + x$	
		sage:	

decryption always worl

beffs of a are in $\{-1, 0, 0\}$ beffs of b are in $\{-1, 0\}$

xactly w are nonzero.

sage: M(u,3)
$$x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$$

+ 1
sage: M(conv(c,d),3)
 $x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$
+ 1
sage: conv(M(u,3),d3)
 $x^{6} - x^{5} - x^{4} - 3*x^{3} - x^{2} + x$
 $x - 3$
sage: M(_,3)
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c

26

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of *b* are in $\{-1, 0, 1\}$, and exactly w are nonzero.

sage: M(u,3)
$$x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$$

+ 1
sage: M(conv(c,d),3)
 $x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$
+ 1
sage: conv(M(u,3),d3)
 $x^{6} - x^{5} - x^{4} - 3*x^{3} - x^{2} + x$
 $x - 3$
sage: M(_,3)
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of *b* are in $\{-1, 0, 1\}$, and exactly w are nonzero.

26

Each coeff of *ab* in *R* has absolute value at most w.

sage: M(u,3)
$$x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$$

+ 1
sage: M(conv(c,d),3)
 $x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$
+ 1
sage: conv(M(u,3),d3)
 $x^{6} - x^{5} - x^{4} - 3*x^{3} - x^{2} + x$
 $x - 3$
sage: M(_,3)
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c

26

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of *b* are in $\{-1, 0, 1\}$, and exactly w are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.)

sage: M(u,3)
$$x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$$

+ 1
sage: M(conv(c,d),3)
 $x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$
+ 1
sage: conv(M(u,3),d3)
 $x^{6} - x^{5} - x^{4} - 3*x^{3} - x^{2} + x$
 $x - 3$
sage: M(_,3)
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c

26

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of *b* are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of ab in Rhas absolute value at most w. (Same argument would work for

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

- b of any weight, a of weight w.)

sage: M(u,3)
$$x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$$

+ 1
sage: M(conv(c,d),3)
 $x^{6} - x^{5} + x^{4} - x^{3} - x^{2} - x$
+ 1
sage: conv(M(u,3),d3)
 $x^{6} - x^{5} - x^{4} - 3*x^{3} - x^{2} + x$
 $x - 3$
sage: M(_,3)
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c
 $x^{6} - x^{5} - x^{4} - x^{2} + x$
sage: c

26

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096.

- b of any weight, a of weight w.)

26								
	X		X		+			
	_		_		2			
	2		2		x			
	x		x		_		X	
	_		-		3		+	
	3		3)	x		2	
	x	3)	x^	d3	3*		x^	
	-	l),	_	3),	_		-	
	`4	с, с	`4	1,3	`4		`4	
	Х́	7(0	Х́	1(1	Х́)	Х́	
	+	n	+	7 (N	_	,3)	_	
(u,	^5	(cc	^5	onv	^5	(_ ;	^5	

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.)

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096.

What at

$$- x^{3} - x^{2} - x$$

$$),3)$$

$$- x^{3} - x^{2} - x$$

$$),d3)$$

$$- 3*x^{3} - x^{2} + x$$

$$- x^{2} + x$$

26

Does decryption always work?

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.) Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096.

What about w = -

^2 - x

-^2 - x

x^2 +

26

Does decryption always work?

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.) Similar comments for d, c.

Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096.

27

What about w = 467, q = 2

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of *b* are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.)

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096.

What about w = 467, q = 2048?

All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.)

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096. 27

What about w = 467, q = 2048?

Same argument doesn't work.

a = b = c = d =

 $1 + x + x^2 + \cdots + x^{w-1}$:

3ab + dc has a coeff 4w > q/2.



All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.)

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096. 27

What about w = 467, q = 2048?

Same argument doesn't work.

a = b = c = d =

 $1 + x + x^2 + \cdots + x^{w-1}$:

3ab + dc has a coeff 4w > q/2.

But coeffs are usually <1024when *a*, *d* are chosen randomly.



All coeffs of a are in $\{-1, 0, 1\}$. All coeffs of b are in $\{-1, 0, 1\}$, and exactly *w* are nonzero.

Each coeff of *ab* in *R* has absolute value at most w. (Same argument would work for b of any weight, a of weight w.)

Similar comments for d, c. Each coeff of 3ab + dc in R has absolute value at most 4w.

e.g. w = 467: at most 1868. Decryption works for q = 4096.

What about w = 467, q = 2048? Same argument doesn't work. a = b = c = d = $1 + x + x^2 + \cdots + x^{w-1}$: 3ab + dc has a coeff 4w > q/2. But coeffs are usually <1024when *a*, *d* are chosen randomly. 1996 NTRU handout mentioned no-decryption-failure option, but recommended smaller q with some chance of failures. 1998 NTRU paper: decryption failure "will occur so rarely that it can be ignored in practice".

cryption always work?

is of *a* are in $\{-1, 0, 1\}$. is of *b* are in $\{-1, 0, 1\}$, itly *w* are nonzero.

eff of *ab* in *R* olute value at most *w*. rgument would work for weight, *a* of weight *w*.)

comments for *d*, *c*.

eff of 3ab + dc in R

olute value at most 4w.

= 467: at most 1868. Ion works for q = 4096. What about w = 467, q = 2048?

27

Same argument doesn't work. a = b = c = d = $1 + x + x^2 + \dots + x^{w-1}$: 3ab + dc has a coeff 4w > q/2.

But coeffs are usually <1024 when *a*, *d* are chosen randomly.

1996 NTRU handout mentioned no-decryption-failure option, but recommended smaller *q* with some chance of failures. 1998 NTRU paper: decryption failure "will occur so rarely that it can be ignored in practice".

Crypto 2 Nguyen-Silverma "The im decrypti security Decrypt "all the for vario not be v

lways work?

27

in $\{-1, 0, 1\}$. in $\{-1, 0, 1\}$,

nonzero.

n *R*

at most *w*. vould work for

of weight w.)

for *d*, *c*.

+ dc in R

at most 4w.

most 1868.

for q = 4096.

What about w = 467, q = 2048?

Same argument doesn't work. a = b = c = d = $1 + x + x^2 + \dots + x^{w-1}$: 3ab + dc has a coeff 4w > q/2. But coeffs are usually <1024when *a*, *d* are chosen randomly. 1996 NTRU handout mentioned no-decryption-failure option, but recommended smaller q with some chance of failures. 1998 NTRU paper: decryption failure "will occur so rarely that it can be ignored in practice".

Crypto 2003 Howg Nguyen–Pointchev Silverman–Singer– "The impact of decryption failures security of NTRU Decryption failures "all the security p for various NTRU

not be valid after

<u><?</u> 1}. ,1}, 27

N. for w.)

? 1*w*.

.)96.

Same argument doesn't work. a = b = c = d = $1 + x + x^2 + \cdots + x^{w-1}$: 3ab + dc has a coeff 4w > q/2. But coeffs are usually <1024when *a*, *d* are chosen randomly. 1996 NTRU handout mentioned no-decryption-failure option, but recommended smaller q with some chance of failures. 1998 NTRU paper: decryption failure "will occur so rarely that it can be ignored in practice".

What about w = 467, q = 2048?

28

Crypto 2003 Howgrave-Grah Nguyen–Pointcheval–Proos–

- Silverman–Singer–Whyte
- "The impact of
- decryption failures on the
- security of NTRU encryption
- Decryption failures imply the
- "all the security proofs know
- for various NTRU paddings
- not be valid after all".

What about w = 467, q = 2048?

Same argument doesn't work. a = b = c = d = $1 + x + x^2 + \cdots + x^{w-1}$: 3ab + dc has a coeff 4w > q/2.

But coeffs are usually <1024when *a*, *d* are chosen randomly.

1996 NTRU handout mentioned no-decryption-failure option, but recommended smaller q with some chance of failures. 1998 NTRU paper: decryption failure "will occur so rarely that it can be ignored in practice".

Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption": Decryption failures imply that "all the security proofs known ...

28

not be valid after all".

- for various NTRU paddings may

What about w = 467, q = 2048?

Same argument doesn't work. a = b = c = d = $1 + x + x^2 + \cdots + x^{w-1}$: 3ab + dc has a coeff 4w > q/2.

But coeffs are usually <1024when *a*, *d* are chosen randomly.

1996 NTRU handout mentioned no-decryption-failure option, but recommended smaller q with some chance of failures. 1998 NTRU paper: decryption failure "will occur so rarely that it can be ignored in practice".

Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption": Decryption failures imply that "all the security proofs known ... for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!

pout w = 467, q = 2048?

28

gument doesn't work.

c = d = $x^2 + \cdots + x^{w-1}$: c has a coeff 4w > q/2.

ffs are usually <1024d are chosen randomly.

RU handout mentioned ption-failure option, mmended smaller q ne chance of failures. ⁻RU paper: decryption will occur so rarely that e ignored in practice".

Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption":

Decryption failures imply that "all the security proofs known ... for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!

29

Coeff of $c_0 d_{n-1}$ -This coe *C*₀, *C*₁, . . high cor

 d_{n-1} , d_n

467, *q* = 2048?

28

pesn't work.

- $-x^{w-1}$:
- oeff 4w > q/2.
- ally <1024 sen randomly.
- out mentioned
- ire option,
- smaller q
- of failures.
- r: decryption
- so rarely that
- n practice".

Crypto 2003 Howgrave-Graham– Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption":

Decryption failures imply that "all the security proofs known . . . for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!

Coeff of x^{n-1} in c_{0} $c_{0}d_{n-1} + c_{1}d_{n-2} - c_{0}$ This coeff is large $c_{0}, c_{1}, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_{0}$

2048?

28

k.

q/2.

4 nly.

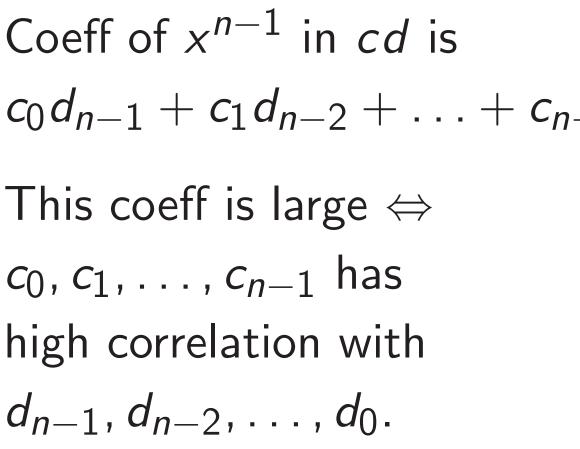
oned

` **`** on that

,,

Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption": Decryption failures imply that "all the security proofs known ... for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!



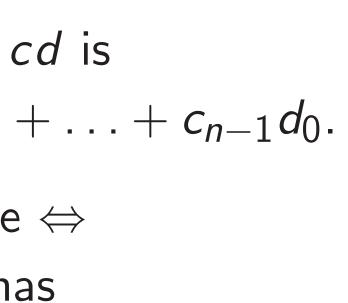
Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption":

Decryption failures imply that "all the security proofs known for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$

29



Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption":

Decryption failures imply that "all the security proofs known . . . for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, ..., c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$.

29

Crypto 2003 Howgrave-Graham-Nguyen–Pointcheval–Proos– Silverman–Singer–Whyte "The impact of decryption failures on the security of NTRU encryption":

Decryption failures imply that "all the security proofs known ... for various NTRU paddings may not be valid after all".

Even worse: Attacker who sees some random decryption failures can figure out the secret key!

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$. i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$

29

2003 Howgrave-Graham--Pointcheval–Proos–

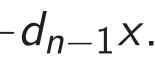
29

- n–Singer–Whyte
- pact of
- on failures on the
- of NTRU encryption":
- on failures imply that security proofs known ... us NTRU paddings may alid after all".
- orse: Attacker who sees ndom decryption failures re out the secret key!

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$.

i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$

Reasona random c correla



grave-Graham val—Proos— ·Whyte 29

- on the encryption":
- s imply that roofs known ...
- paddings may all".
- ker who sees yption failures secret key!

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, ..., c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$. i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where

 $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$

Reasonable guesse random decryption *c* correlated with s

nam–

29

า":

at /n . . . may

ees lures /!

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$. i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where

 $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$

Reasonable guesses given a random decryption failure: c correlated with some x^i re

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, ..., c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$. i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where

 $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$.

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, ..., c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$. i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where

 $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$

30

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$.

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, ..., c_{n-1}$ has high

correlation with some rotation

of $d_{n-1}, d_{n-2}, \ldots, d_0$.

i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$ 30

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$.

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow

 $c_0, c_1, ..., c_{n-1}$ has high correlation with some rotation of $d_{n-1}, d_{n-2}, \ldots, d_0$.

i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$ 30

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$. Experimentally confirmed:

Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$.

Coeff of x^{n-1} in *cd* is $c_0 d_{n-1} + c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ This coeff is large \Leftrightarrow $c_0, c_1, \ldots, c_{n-1}$ has high correlation with $d_{n-1}, d_{n-2}, \ldots, d_0.$ Some coeff is large \Leftrightarrow $c_0, c_1, ..., c_{n-1}$ has high correlation with some rotation

of $d_{n-1}, d_{n-2}, \ldots, d_0$.

i.e. c is correlated with $x' \operatorname{rev}(d)$ for some *i*, where $rev(d) = d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$ 30

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$. Experimentally confirmed: Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$. Eurocrypt 2002 Gentry–Szydlo

31

algorithm then finds d.

 x^{n-1} in *cd* is $+ c_1 d_{n-2} + \ldots + c_{n-1} d_0.$ eff is large \Leftrightarrow ., c_{n-1} has relation with $_{-2}, \ldots, d_0.$ beff is large \Leftrightarrow ., c_{n-1} has high on with some rotation $d_{n-2}, \ldots, d_0.$ correlated with

) for some *i*, where $= d_0 + d_1 x^{n-1} + \cdots + d_{n-1} x.$ Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$.

30

Experimentally confirmed: Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$.

Eurocrypt 2002 Gentry–Szydlo algorithm then finds d.

31

1999 Ha 2000 Ja Hoffstei Fluhrer, using inv

d is $+ \ldots + c_{n-1} d_0.$ \Leftrightarrow IS ith $e \Leftrightarrow$ is high me rotation d_0 . with

30

i, where

 $b^{n-1}+\cdots+d_{n-1}x$.

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$. Experimentally confirmed: Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$. Eurocrypt 2002 Gentry–Szydlo

algorithm then finds d.

1999 Hall–Goldber 2000 Jaulmes–Jou Hoffstein–Silverma Fluhrer, etc.: Ever using invalid mess

 $_{-1}d_{0}$.

30

Reasonable guesses given a random decryption failure: c correlated with some $x' \operatorname{rev}(d)$. rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$. Experimentally confirmed: Average of $c \operatorname{rev}(c)$

over some decryption failures

is close to $d \operatorname{rev}(d)$.

Round to integers: $d \operatorname{rev}(d)$.

Eurocrypt 2002 Gentry–Szydlo algorithm then finds d.

31

on

 $d_{n-1}x$.

1999 Hall–Goldberg–Schneie 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier at using invalid messages.

Reasonable guesses given a random decryption failure: c correlated with some x^{i} rev(d). rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$.

Experimentally confirmed: Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$.

Eurocrypt 2002 Gentry–Szydlo algorithm then finds d.

31

1999 Hall–Goldberg–Schneier, 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier attacks using invalid messages.

Reasonable guesses given a random decryption failure: c correlated with some x^{i} rev(d). rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$.

Experimentally confirmed: Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$.

Eurocrypt 2002 Gentry–Szydlo algorithm then finds d.

31

1999 Hall–Goldberg–Schneier, 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier attacks using invalid messages. Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

Reasonable guesses given a random decryption failure: c correlated with some x^{i} rev(d). rev(c) correlated with $x^{-i}d$. $c \operatorname{rev}(c)$ correlated with $d \operatorname{rev}(d)$.

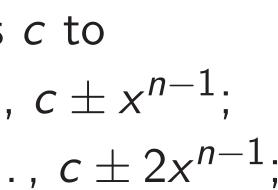
Experimentally confirmed: Average of $c \operatorname{rev}(c)$ over some decryption failures is close to $d \operatorname{rev}(d)$. Round to integers: $d \operatorname{rev}(d)$.

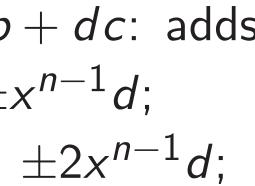
Eurocrypt 2002 Gentry–Szydlo algorithm then finds d.

31

1999 Hall–Goldberg–Schneier, 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier attacks using invalid messages. Attacker changes c to $c \pm 1, c \pm x, \ldots, c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc. This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$; $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d;$

 $\pm 3d$, etc.





ble guesses given a decryption failure: ated with some $x' \operatorname{rev}(d)$. orrelated with $x^{-\prime}d$. correlated with $d \operatorname{rev}(d)$. 31

entally confirmed:

- of $c \operatorname{rev}(c)$
- ne decryption failures
- to $d \operatorname{rev}(d)$.
- o integers: $d \operatorname{rev}(d)$.

ot 2002 Gentry–Szydlo n then finds d.

1999 Hall–Goldberg–Schneier, 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier attacks using invalid messages.

Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$; $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d;$ $\pm 3d$, etc.

32

e.g. 3*ab* all other and d =

is given a in failure: some $x^i \operatorname{rev}(d)$. with $x^{-i}d$. with $d \operatorname{rev}(d)$. 31

nfirmed:

ion failures).

 $d \operatorname{rev}(d)$.

entry–Szydlo ds *d*. 1999 Hall–Goldberg–Schneier, 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier attacks using invalid messages.

Attacker changes c to $c \pm 1, c \pm x, \ldots, c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3,$ etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$; $\pm 2d$, $\pm 2xd$, ..., $\pm 2x^{n-1}d$; $\pm 3d$, etc.

e.g. $3ab+dc = \cdots$ all other coeffs in and $d = \cdots + x^{47}$

31

v(d).

v(d).

S

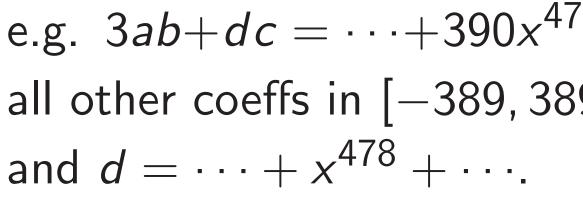
olb

1999 Hall–Goldberg–Schneier, 2000 Jaulmes–Joux, 2000 Hoffstein–Silverman, 2016 Fluhrer, etc.: Even easier attacks using invalid messages.

Attacker changes c to $c \pm 1, c \pm x, \ldots, c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$; $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d;$ $\pm 3d$, etc.

32

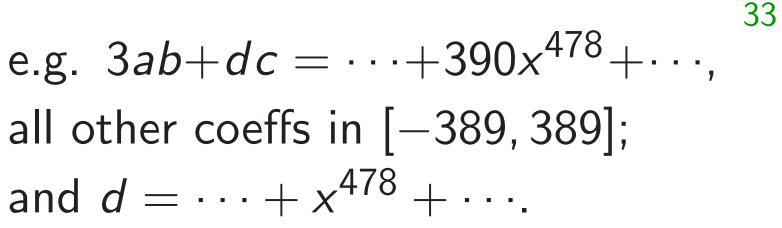


Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1}$: $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$: $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d$: $\pm 3d$, etc.

32

all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$.

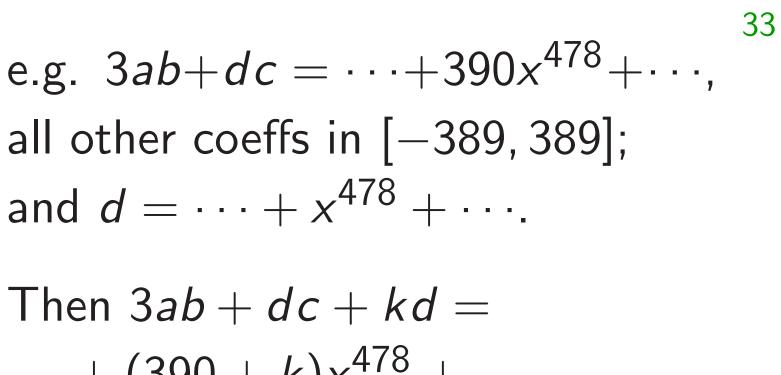


Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1}$: $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$: $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d;$ $\pm 3d$, etc.

all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$. Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$

32



Decryption fails for big k.

Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$: $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d$: $\pm 3d$, etc.

all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$. Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k.

32

Search for smallest k that falis.

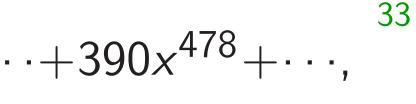
33 e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$,

Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$: $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d;$ $\pm 3d$, etc.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$ Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k. Search for smallest k that falis. Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$. i.e., if $d = \cdots + x^{477} + \cdots$

32



Attacker changes c to $c \pm 1, c \pm x, ..., c \pm x^{n-1};$ $c \pm 2, c \pm 2x, \ldots, c \pm 2x^{n-1};$ $c \pm 3$, etc.

This changes 3ab + dc: adds $\pm d$, $\pm xd$, ..., $\pm x^{n-1}d$: $\pm 2d, \pm 2xd, \ldots, \pm 2x^{n-1}d;$ $\pm 3d$, etc.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$ Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k. Search for smallest k that falis. Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$. i.e., if $d = \cdots + x^{477} + \cdots$ Try x^2kd , x^3kd , etc. See pattern of *d* coeffs.

32

33

II–Goldberg–Schneier, ulmes–Joux, 2000 n–Silverman, 2016 etc.: Even easier attacks valid messages.

32

^r changes *c* to $\pm x, \ldots, c \pm x^{n-1};$ $\pm 2x, \ldots, c \pm 2x^{n-1};$

tc.

nges 3ab + dc: adds $d, \ldots, \pm x^{n-1}d;$ $2xd, ..., \pm 2x^{n-1}d;$ С.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$. Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k. Search for smallest k that falis. Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$, i.e., if $d = \cdots + x^{477} + \cdots$. Try x^2kd , x^3kd , etc. See pattern of *d* coeffs.

33

How to

Approac

constant

For each

generate Use sign that not rg–Schneier, x, 2000 an, 2016 n easier attacks ages. 32

c to $c \pm x^{n-1};$, $c \pm 2x^{n-1};$

+ dc: adds $x^{n-1}d;$ $\pm 2x^{n-1}d;$ e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$. Then 3ab + dc + kd = $\cdots + (390+k)x^{478} + \cdots$ Decryption fails for big k. Search for smallest k that falis. Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$. i.e., if $d = \cdots + x^{477} + \cdots$. Try x^2kd , x^3kd , etc. See pattern of *d* coeffs.

How to handle inv

Approach 1: Tell I constantly switch

For each new send

generate new publ

Use signatures to that nobody else u

er,

32

tacks

S

-1.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$.

Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k. Search for smallest k that falis. Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$, i.e., if $d = \cdots + x^{477} + \cdots$. Try x^2kd , x^3kd , etc.

See pattern of *d* coeffs.

33

How to handle invalid messa

- Approach 1: Tell user to constantly switch keys.
- For each new sender,
- generate new public key.
- Use signatures to ensure that nobody else uses key.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$.

Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k.

Search for smallest k that falis.

Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$. i.e., if $d = \cdots + x^{477} + \cdots$.

Try x^2kd , x^3kd , etc. See pattern of *d* coeffs. How to handle invalid messages

33

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$.

Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k.

Search for smallest k that falis.

Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{478} + \cdots$, i.e., if $d = \cdots + x^{477} + \cdots$.

Try x^2kd , x^3kd , etc. See pattern of *d* coeffs. How to handle invalid messages

33

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

e.g. $3ab+dc = \cdots + 390x^{478} + \cdots$, all other coeffs in [-389, 389]; and $d = \cdots + x^{478} + \cdots$.

Then 3ab + dc + kd = $\cdots + (390 + k)x^{478} + \cdots$ Decryption fails for big k.

Search for smallest k that falis.

Does 3ab + dc + kxd also fail? Yes if $xd = \cdots + x^{4/8} + \cdots$, i.e., if $d = \cdots + x^{477} + \cdots$.

Try x^2kd , x^3kd , etc. See pattern of *d* coeffs. How to handle invalid messages

33

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks.

 $y+dc=\cdots+390x^{478}+\cdots,$ coeffs in [-389, 389]; $\cdots + x^{478} + \cdots$

33

b + dc + kd = $90 + k x^{478} + \cdots$ ion fails for big k.

or smallest k that falis.

b + dc + kxd also fail? $d=\cdots+x^{478}+\cdots.$ $= \cdots + x^{477} + \cdots$

 $d, x^3 k d$, etc. ern of *d* coeffs.

How to handle invalid messages

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks.

Approac encrypti eliminat

 $+390x^{478}+\cdots$ [-389, 389]; $8 + \cdots$ kd =

33

- $78 + \cdots$
- or big k.
- t k that falis.

kxd also fail? $x^{478} + \cdots$ $477 + \cdots$

etc.

coeffs.

How to handle invalid messages

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks.

Approach 2: Mod encryption and de eliminate invalid n

⁸+···, 9];

33

alis. fail?

• ,

How to handle invalid messages

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks.

34

Approach 2: Modify encryption and decryption to eliminate invalid messages.

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks. 34

Approach 2: Modify encryption and decryption to eliminate invalid messages.

35

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks. 34

Approach 2: Modify encryption and decryption to eliminate invalid messages.

e.g. "IND-CCA" New Hope submission; most submissions.

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks.

Approach 2: Modify encryption and decryption to eliminate invalid messages. e.g. "IND-CCA" New Hope submission; most submissions. Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

34

Approach 1: Tell user to constantly switch keys.

For each new sender, generate new public key. Use signatures to ensure that nobody else uses key.

e.g. original "IND-CPA" version of New Hope; Ding.

If user reuses a key: Blame user for the attacks.

Approach 2: Modify encryption and decryption to eliminate invalid messages. e.g. "IND-CCA" New Hope submission; most submissions. Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message. But encryption is randomized!

34

Reencryption won't match.

handle invalid messages

34

h 1: Tell user to ly switch keys.

- new sender,
- e new public key.
- atures to ensure
- ody else uses key.
- inal "IND-CPA" version Hope; Ding.
- euses a key:
- ser for the attacks.

Approach 2: Modify encryption and decryption to eliminate invalid messages.

e.g. "IND-CCA" New Hope submission; most submissions.

Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

But encryption is randomized! Reencryption won't match.

Solution all rando

e.g. afte compute

alid messages

34

user to

keys.

ler,

ic key.

ensure

ises key.

-CPA" version

g.

y:

e attacks.

Approach 2: Modify encryption and decryption to eliminate invalid messages. e.g. "IND-CCA" New Hope submission; most submissions. Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches

reencryption of message.

But encryption is randomized! Reencryption won't match.

Solution: In decry all randomness that

e.g. after computing compute *b* from 3

ages

34

sion

Approach 2: Modify encryption and decryption to eliminate invalid messages.

e.g. "IND-CCA" New Hope submission; most submissions.

Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

But encryption is randomized! Reencryption won't match.

35

e.g. after computing c in N⁻ compute *b* from 3ab + dc.

Solution: In decryption, con all randomness that was use

e.g. "IND-CCA" New Hope submission; most submissions.

Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

But encryption is randomized! Reencryption won't match.

35

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute b from 3ab + dc.

e.g. "IND-CCA" New Hope submission; most submissions.

Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

But encryption is randomized! Reencryption won't match.

35

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute b from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

e.g. "IND-CCA" New Hope submission; most submissions.

Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

But encryption is randomized! Reencryption won't match.

35

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute b from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

e.g. "IND-CCA" New Hope submission; most submissions.

Basic idea, from Crypto 1999 Fujisaki–Okamoto: After decrypting message, check whether (1) message is valid and (2) ciphertext matches reencryption of message.

But encryption is randomized! Reencryption won't match.

Solution: In decryption, compute all randomness that was used.

35

e.g. after computing c in NTRU, compute *b* from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

h 2: Modify on and decryption to e invalid messages.

D-CCA" New Hope on; most submissions.

ea, from Crypto 1999 -Okamoto: After

ng message, check

(1) message is valid ciphertext matches tion of message.

ryption is randomized! ption won't match.

Solution: In decryption, compute all randomness that was used.

35

e.g. after computing c in NTRU, compute *b* from 3ab + dc.

Can view (b, c) as message, no further randomness.

"Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

How to

36

Eliminat not enoi using de random

ify

cryption to

35

nessages.

lew Hope submissions.

Crypto 1999

- : After
- e, check

ige is valid

matches

essage.

randomized!

't match.

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute b from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

How to handle dee

Eliminating invalid not enough: reme using decryption f random valid mess

35

IS.

9

d!

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute *b* from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

36

How to handle decryption fa

- Eliminating invalid messages
- not enough: remember atta
- using decryption failures for
- random valid messages.

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute *b* from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

36

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute *b* from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

36

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

NIST encryption submissions vary in failure rates.

NTRU HRSS, NTRU Prime, Odd Manhattan choose q to eliminate decryption failures.

Solution: In decryption, compute all randomness that was used.

e.g. after computing c in NTRU, compute *b* from 3ab + dc.

Can view (b, c) as message, no further randomness. "Deterministic encryption."

"Product NTRU" variant is not naturally deterministic.

Generic Fujisaki–Okamoto solution: Require sender to compute randomness as standard hash of message.

36

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

NIST encryption submissions vary in failure rates.

NTRU HRSS, NTRU Prime, Odd Manhattan choose q to eliminate decryption failures.

LIMA tried to eliminate decryption failures, but failed.

: In decryption, compute omness that was used.

36

r computing c in NTRU, b from 3ab + dc.

v(b, c) as message, er randomness. inistic encryption."

t NTRU" variant aturally deterministic.

Fujisaki–Okamoto

- Require sender to
- e randomness as

hash of message.

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

NIST encryption submissions vary in failure rates.

NTRU HRSS, NTRU Prime, Odd Manhattan choose q to eliminate decryption failures.

LIMA tried to eliminate decryption failures, but failed.

37

More cla LOTUS: New Ho KINDI: 2 NTRUE KCL: ≈ 2 Ding: \approx Current what de

is small decrypti were cal ption, compute at was used. 36

ng c in NTRU, ab + dc.

message,

ness. cryption."

variant

terministic.

)kamoto

sender to

ess as

nessage.

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

NIST encryption submissions vary in failure rates.

NTRU HRSS, NTRU Prime, Odd Manhattan choose *q* to eliminate decryption failures.

LIMA tried to eliminate decryption failures, but failed.

More claimed failu LOTUS: $<2^{-256}$. New Hope submiss KINDI: 2^{-165} . : NTRUEncrypt: <100KCL: $\approx 2^{-60}$.

Ding: $\approx 2^{-60}$, only Current debates al

what decryption fa

is small enough; w decryption failure

were calculated co

npute d.

36

ΓRU,

2.

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

NIST encryption submissions vary in failure rates.

NTRU HRSS, NTRU Prime, Odd Manhattan choose q to eliminate decryption failures.

LIMA tried to eliminate decryption failures, but failed. KINDI: 2^{-165} .

37

NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA

Current debates about what decryption failure prob is small enough; whether decryption failure probabiliti were calculated correctly; et

More claimed failure rates: LOTUS: $<2^{-256}$.

New Hope submission: $<2^{-}$

How to handle decryption failures

Eliminating invalid messages is not enough: remember attack using decryption failures for random valid messages.

NIST encryption submissions vary in failure rates.

NTRU HRSS, NTRU Prime, Odd Manhattan choose q to eliminate decryption failures.

LIMA tried to eliminate decryption failures, but failed.

More claimed failure rates: LOTUS: $<2^{-256}$. New Hope submission: $<2^{-213}$. KINDI: 2^{-165} .

NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA.

37

Current debates about what decryption failure probability is small enough; whether decryption failure probabilities were calculated correctly; etc.

handle decryption failures

37

ing invalid messages is ugh: remember attack cryption failures for valid messages.

cryption submissions ailure rates.

IRSS, NTRU Prime, nhattan choose q to e decryption failures.

ied to eliminate on failures, but failed.

More claimed failure rates: LOTUS: $<2^{-256}$. New Hope submission: $<2^{-213}$. KINDI: 2^{-165} . NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA.

Current debates about what decryption failure probability is small enough; whether decryption failure probabilities were calculated correctly; etc.



How to

38

If messa Attacker a guess

cryption failures

37

l messages is mber attack ailures for

sages.

ubmissions

s.

RU Prime,

hoose q to

on failures.

ninate

, but failed.

More claimed failure rates: LOTUS: $<2^{-256}$. New Hope submission: $<2^{-213}$. KINDI: 2^{-165} . NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA. Current debates about what decryption failure probability is small enough; whether decryption failure probabilities

were calculated correctly; etc.

How to randomize

If message is guess Attacker can chec a guess matches a

	37
ilures	
s is	
ck	
5	
•	
7	
d.	

NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA.

Current debates about what decryption failure probability is small enough; whether decryption failure probabilities were calculated correctly; etc.

38

How to randomize messages

If message is guessable:

Attacker can check whether

a guess matches a ciphertex

NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA.

Current debates about what decryption failure probability is small enough; whether decryption failure probabilities were calculated correctly; etc.

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA.

Current debates about what decryption failure probability is small enough; whether decryption failure probabilities were calculated correctly; etc.

38

How to randomize messages

If message is guessable: Attacker can check whether

Also various attacks using guesses of portion of message.

- a guess matches a ciphertext.

NTRUEncrypt: $<2^{-80}$. KCL: $\approx 2^{-60}$. Ding: $\approx 2^{-60}$, only IND-CPA.

Current debates about what decryption failure probability is small enough; whether decryption failure probabilities were calculated correctly; etc.

38

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data.

aimed failure rates:

 $< 2^{-256}$

pe submission: $<2^{-213}$. 2^{-165}

ncrypt: $< 2^{-80}$. 2^{-60} 2^{-60} , only IND-CPA.

debates about cryption failure probability enough; whether on failure probabilities culated correctly; etc.

How to randomize messages

38

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data.

Central Can atta a randor public ke

39

ire rates:

sion: $<2^{-213}$.

38

 2^{-80} .

/ IND-CPA.

bout

ailure probability

hether

probabilities

rrectly; etc.

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data.

Central "one-wayr Can attacker figur a random message public key and cip

38

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data. Central "one-wayness" ques Can attacker figure out a random message given public key and ciphertext?

39

ability

213

es

C.

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data.

39

Central "one-wayness" question: Can attacker figure out a random message given public key and ciphertext?

40

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data.

39

Central "one-wayness" question: Can attacker figure out a random message given public key and ciphertext? Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

How to randomize messages

If message is guessable: Attacker can check whether a guess matches a ciphertext.

Also various attacks using guesses of portion of message.

Modern "KEM-DEM" solution, from Eurocrypt 2000 Shoup: Choose random message. Use hash of message as (e.g.) AES-256-GCM key to encrypt and authenticate user data.

Central "one-wayness" question: Can attacker figure out a random message given public key and ciphertext? Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness. Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

39

randomize messages

- ge is guessable: can check whether matches a ciphertext.
- ious attacks using of portion of message.
- "KEM-DEM" solution, rocrypt 2000 Shoup: random message. n of message as (e.g.)
- 6-GCM key to encrypt nenticate user data.

Central "one-wayness" question: Can attacker figure out a random message given public key and ciphertext?

39

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

40

Brute-fo

Attacker A = 3a/Can atta

messages

39

sable:

k whether

ciphertext.

ks using

of message.

EM" solution,

00 Shoup:

essage.

ge as (e.g.)

y to encrypt user data. Central "one-wayness" question: Can attacker figure out a random message given public key and ciphertext?

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

Brute-force search

Attacker is given p A = 3a/d, ciphert Can attacker find

39

t.

ςe.

on,

; .) pt Central "one-wayness" question: Can attacker figure out a random message given public key and ciphertext?

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

40

Brute-force search

Attacker is given public key A = 3a/d, ciphertext C = ACan attacker find *c*?

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

40

Brute-force search

Attacker is given public key Can attacker find *c*?

A = 3a/d, ciphertext C = Ab + c.

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

40

Brute-force search

Attacker is given public key Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

A = 3a/d, ciphertext C = Ab + c.

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

40

Brute-force search

Attacker is given public key Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different also stop legitimate decryption.)

- A = 3a/d, ciphertext C = Ab + c.
- messages c? Unlikely. This would

Fujisaki–Okamoto and many newer papers try to prove that all chosen-ciphertext distinguishers ("IND-CCA attacks") are as difficult as breaking one-wayness.

Many limitations to proofs: bugs; looseness; assumptions of "ROM" or "QROM" attacks; assumptions on failure probability; etc.

40

Brute-force search

Attacker is given public key Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different also stop legitimate decryption.)

Or search 3^n choices of d. decrypt. Slightly slower but can be reused for many ciphertexts.

- A = 3a/d, ciphertext C = Ab + c.
- messages c? Unlikely. This would
- If a = dA/3 is small, use (a, d) to

"one-wayness" question: acker figure out n message given ey and ciphertext?

-Okamoto and many apers try to prove that all ciphertext distinguishers CA attacks") are as as breaking one-wayness.

nitations to proofs: bugs; s; assumptions of "ROM" OM" attacks; assumptions e probability; etc.

Brute-force search

40

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivale

Secret k secret ke secret ke

less" question:

40

e out

- e given
- hertext?
- and many
- o prove that all distinguishers (s") are as
- g one-wayness.

tions of "ROM" ks; assumptions ity; etc.

Brute-force search

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of *b*. If c = C - Ab is small: done!

(Can this find two different messages *c*? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

Equivalent keys

Secret key (a, d) is secret key (xa, xd)secret key (x^2a, x^2)

tion:

40

nat all ners

ness.

bugs; ROM"

ptions

Brute-force search

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivalent keys

Secret key (a, d) is equivalent secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

n = 701, w = 467:

$\binom{n}{w} 2^{w} \approx 2^{1106.09};$ $3^{n} \approx 2^{1111.06};$ $3^n/n \approx 2^{1101.61}$.

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

n = 701, w = 467:

Exercise: Find more equivalences!

$\binom{n}{w}2^w\approx 2^{1106.09};$ $3^n \approx 2^{1111.06}$: $3^n/n \approx 2^{1101.61}$.

Attacker is given public key A = 3a/d, ciphertext C = Ab + c. Can attacker find *c*?

Search $\binom{n}{w} 2^{w}$ choices of b. If c = C - Ab is small: done!

(Can this find two different messages c? Unlikely. This would also stop legitimate decryption.)

Or search 3^n choices of d. If a = dA/3 is small, use (a, d) to decrypt. Slightly slower but can be reused for many ciphertexts.

41

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

n = 701, w = 467:

Exercise: Find more equivalences!

But if w is chosen smaller then $\binom{n}{w} 2^{w}$ search will be faster.

$\binom{n}{w} 2^{w} \approx 2^{1106.09};$ $3^n \approx 2^{1111.06}$:

 $3^n/n \approx 2^{1101.61}$.

rce search

r is given public key d, ciphertext C = Ab + c. acker find *c*?

 $\binom{n}{w} 2^{w}$ choices of b. -Ab is small: done!

s find two different s c? Unlikely. This would > legitimate decryption.)

h 3^n choices of d. A/3 is small, use (a, d) to Slightly slower but can d for many ciphertexts.

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

n = 701, w = 467: $\binom{n}{w} 2^{w} \approx 2^{1106.09};$ $3^{n} \approx 2^{1111.06};$ $3^n/n \approx 2^{1101.61}$.

Exercise: Find more equivalences!

But if w is chosen smaller then $\binom{n}{w} 2^{w}$ search will be faster.

41

Collision

42

Write *d* $d_1 = bo$ $d_2 = \operatorname{rer}$

```
41
```

bublic key ext C = Ab + c. c?

ices of b.

mall: done!

different kely. This would e decryption.)

ces of d.

all, use (a, d) to

lower but can

y ciphertexts.

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

n = 701, w = 467: $\binom{n}{w} 2^{w} \approx 2^{1106.09};$ $3^{n} \approx 2^{1111.06};$ $3^{n}/n \approx 2^{1101.61}.$

Exercise: Find more equivalences!

But if w is chosen smaller then $\binom{n}{w} 2^{w}$ search will be faster.

Collision attacks

Write d as $d_1 + d_2$ $d_1 =$ bottom $\lceil n/2$ $d_2 =$ remaining te

41

Equivalent keys

b+c.

e!

would on.)

, *d*) to can xts.

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

n = 701, w = 467: $\binom{n}{w} 2^{w} \approx 2^{1106.09};$ $3^n \approx 2^{1111.06}$: $3^n/n \approx 2^{1101.61}$.

Exercise: Find more equivalences!

But if w is chosen smaller then $\binom{n}{w} 2^{w}$ search will be faster.

Collision attacks

42

Write d as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of d_2 = remaining terms of d.

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

$$n = 701, w = 467:$$

 $\binom{n}{w} 2^{w} \approx 2^{1106.09};$
 $3^{n} \approx 2^{1111.06};$
 $3^{n}/n \approx 2^{1101.61}.$

Exercise: Find more equivalences!

But if w is chosen smaller then $\binom{n}{w}2^{w}$ search will be faster.

42

Collision attacks

Write *d* as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of *d*, $d_2 = \text{remaining terms of } d$. 43

Equivalent keys

Secret key (*a*, *d*) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

$$n = 701, w = 467:$$

 $\binom{n}{w} 2^{w} \approx 2^{1106.09};$
 $3^{n} \approx 2^{1111.06};$
 $3^{n}/n \approx 2^{1101.61}.$

Exercise: Find more equivalences!

But if w is chosen smaller then $\binom{n}{w} 2^{w}$ search will be faster.

Collision attacks

42

Write *d* as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of d, d_2 = remaining terms of d.

$$a = (A/3)d = (A/3)d_2 =$$

so $a - (A/3)d_2 =$

 $(A/3)d_1 + (A/3)d_2$ $= (A/3)d_1.$

Equivalent keys

Secret key (a, d) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

$$n = 701, w = 467:$$

 $\binom{n}{w} 2^{w} \approx 2^{1106.09};$
 $3^{n} \approx 2^{1111.06};$
 $3^{n}/n \approx 2^{1101.61}.$

Exercise: Find more equivalences!

But if *w* is chosen smaller then $\binom{n}{w} 2^{w}$ search will be faster.

Collision attacks

42

Write d as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of d, d_2 = remaining terms of d.

 $a = (A/3)d = (A/3)d_1 + (A/3)d_2$ so $a - (A/3)d_2 = (A/3)d_1$. Eliminate a: almost certainly $H(-(A/3)d_2) = H((A/3)d_1)$ for $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$

Equivalent keys

Secret key (*a*, *d*) is equivalent to secret key (xa, xd), secret key (x^2a, x^2d) , etc.

Search only about $3^n/n$ choices.

$$n = 701, w = 467:$$

 $\binom{n}{w} 2^{w} \approx 2^{1106.09};$
 $3^{n} \approx 2^{1111.06};$
 $3^{n}/n \approx 2^{1101.61}.$

Exercise: Find more equivalences!

But if *w* is chosen smaller then $\binom{n}{w}2^{w}$ search will be faster.

Collision attacks

42

Write d as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of d, d_2 = remaining terms of d.

 $a = (A/3)d = (A/3)d_1 + (A/3)d_2$ so $a - (A/3)d_2 = (A/3)d_1$. Eliminate a: almost certainly $H(-(A/3)d_2) = H((A/3)d_1)$ for $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$

Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory.

nt keys

- ey (a, d) is equivalent to ey (xa, xd), ev $(x^2 a, x^2 d)$, etc.
- only about $3^n/n$ choices.

,
$$w = 467$$
:
 $\binom{n}{w} 2^{w} \approx 2^{1106.09}$;
 $3^{n} \approx 2^{1111.06}$;
 $3^{n}/n \approx 2^{1101.61}$.

- : Find more equivalences!
- ' is chosen smaller then earch will be faster.

Collision attacks

42

Write d as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil \text{ terms of } d$, d_2 = remaining terms of d.

 $a = (A/3)d = (A/3)d_1 + (A/3)d_2$ so $a - (A/3)d_2 = (A/3)d_1$. Eliminate *a*: almost certainly $H(-(A/3)d_2) = H((A/3)d_1)$ for $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$

Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory.

43

Lattices

```
42
```

s equivalent to d^{2} , etc.

 $3^n/n$ choices.

 $\binom{n}{N} 2^{W} \approx 2^{1106.09};$ $3^n \approx 2^{1111.06}$: $3^{n}/n \approx 2^{1101.61}$.

re equivalences!

smaller then be faster.

Collision attacks

Write *d* as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of d, d_2 = remaining terms of d.

 $a = (A/3)d = (A/3)d_1 + (A/3)d_2$ so $a - (A/3)d_2 = (A/3)d_1$. Eliminate a: almost certainly $H(-(A/3)d_2) = H((A/3)d_1)$ for $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$ Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory.

Lattices

nt to

ces.

106.09. 1111.06. 101.61

ences!

hen

Write d as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of d, d_2 = remaining terms of d.

 $a = (A/3)d = (A/3)d_1 + (A/3)d_2$ so $a - (A/3)d_2 = (A/3)d_1$. Eliminate a: almost certainly $H(-(A/3)d_2) = H((A/3)d_1)$ for $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$

Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory.

Lattices

Write *d* as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of *d*, $d_2 = \text{remaining terms of } d$.

$$a = (A/3)d = (A/3)d_1 + (A/3)d_2$$

so $a - (A/3)d_2 = (A/3)d_1$.
Eliminate a : almost certainly
 $H(-(A/3)d_2) = H((A/3)d_1)$ for
 $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$

Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory. 43

Lattices

Write *d* as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of *d*, $d_2 = \text{remaining terms of } d$.

$$a = (A/3)d = (A/3)d_1 + (A/3)d_2$$

so $a - (A/3)d_2 = (A/3)d_1$.
Eliminate a : almost certainly
 $H(-(A/3)d_2) = H((A/3)d_1)$ for
 $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0]).$

Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory.

Lattices

43

This is a lettuce:



Write *d* as $d_1 + d_2$ where $d_1 = \text{bottom } \lceil n/2 \rceil$ terms of *d*, $d_2 = \text{remaining terms of } d$.

$$a = (A/3)d = (A/3)d_1 + (A/3)d_2$$

so $a - (A/3)d_2 = (A/3)d_1$.
Eliminate a : almost certainly
 $H(-(A/3)d_2) = H((A/3)d_1)$ for
 $H(f) = ([f_0 < 0], \dots, [f_{k-1} < 0])$.

Enumerate all $H(-(A/3)d_2)$. Enumerate all $H((A/3)d_1)$. Search for collisions. Only about $3^{n/2}$ computations; but beware cost of memory.

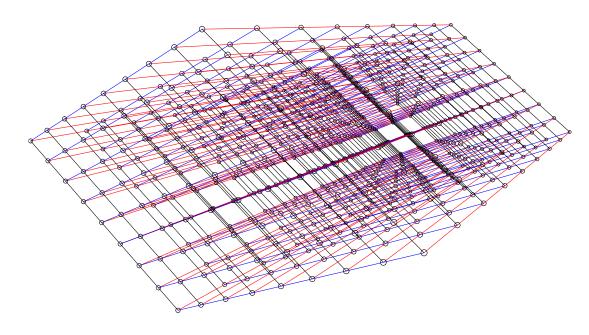
Lattices

43

This is a lettuce:



This is a lattice:



attacks

as $d_1 + d_2$ where ttom $\lceil n/2 \rceil$ terms of d, naining terms of d.

3) $d = (A/3)d_1 + (A/3)d_2$ $A/3)d_2 = (A/3)d_1.$ e a: almost certainly $(3)d_2) = H((A/3)d_1)$ for $([f_0 < 0], \dots, [f_{k-1} < 0]).$

ate all $H(-(A/3)d_2)$. ate all $H((A/3)d_1)$. for collisions.

out $3^{n/2}$ computations;

are cost of memory.

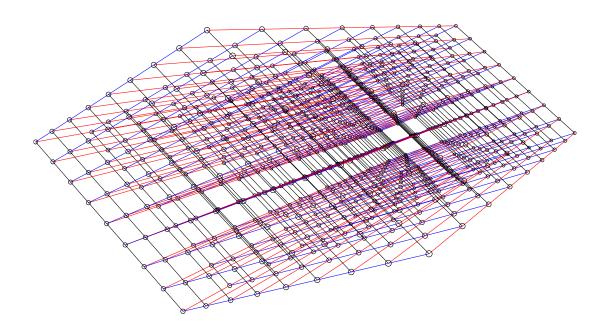
Lattices

43

This is a lettuce:



This is a lattice:



Lattices,

44

Assume are **R**-lin i.e., **R** b_1 $\{r_1b_1 +$ is a *k*-di 43

where where c_1 where d_1 terms of d_1

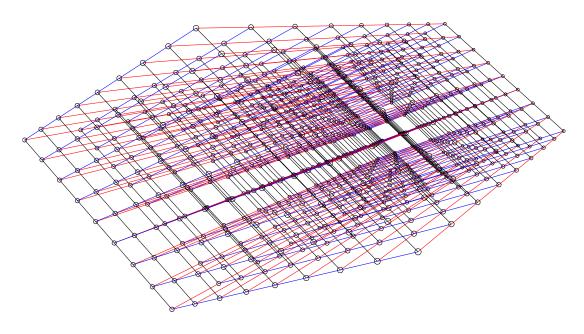
- rms of *d*.
- $(3)d_1 + (A/3)d_2$ $(A/3)d_1.$ st certainly $((A/3)d_1)$ for $\dots, [f_{k-1} < 0]).$
- $-(A/3)d_2).$ $(A/3)d_1).$
- ۱S.
- omputations;
- f memory.

Lattices

This is a lettuce:



This is a lattice:



Lattices, mathema

Assume that b_1 , b_2 are **R**-linearly inder i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}$ $\{r_1b_1 + \ldots + r_kb_k$ is a *k*-dimensional

f*d*,

 $(3)d_2$

) for < 0]).

ns;

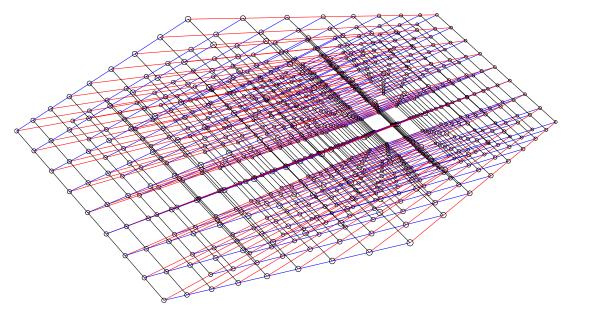
Lattices

43

This is a lettuce:



This is a lattice:



Lattices, mathematically

44

Assume that $b_1, b_2, \ldots, b_k \in$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ ${r_1b_1 + \ldots + r_kb_k : r_1, \ldots, }$

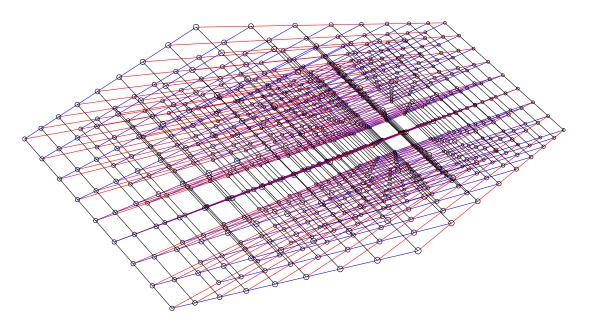
is a k-dimensional vector spa

Lattices

This is a lettuce:



This is a lattice:



44

Lattices, mathematically

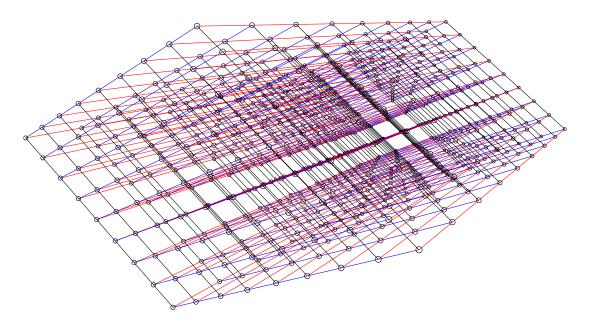
Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a k-dimensional vector space.

Lattices

This is a lettuce:



This is a lattice:



44

Lattices, mathematically

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ is a k-dimensional vector space.

 $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

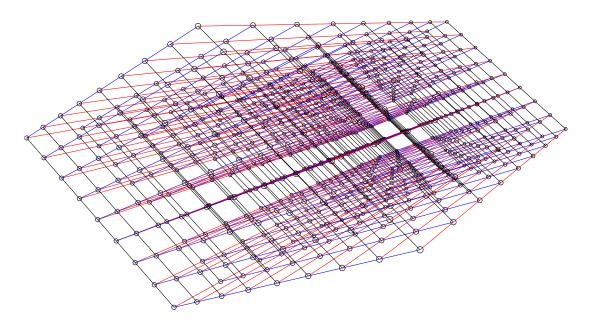
$\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$

Lattices

This is a lettuce:



This is a lattice:



44

Lattices, mathematically

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a k-dimensional vector space.

 $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_k =$ is a rank-k length-n lattice.

 b_1,\ldots,b_k is a **basis** of this lattice.

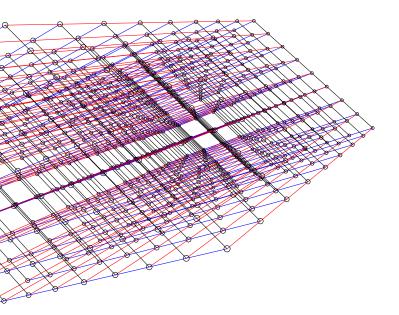
45

$\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$

lettuce:



lattice:



Lattices, mathematically

44

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a k-dimensional vector space.

 $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1, \ldots, b_k is a **basis** of this lattice.

Short ve

45

Given b_1 what is a in $\mathbf{Z}b_1 \dashv$



44

Assume that $b_1, b_2, \ldots, b_k \in \mathbb{R}^n$ are \mathbb{R} -linearly independent, i.e., $\mathbb{R}b_1 + \ldots + \mathbb{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbb{R}\}$ is a k-dimensional vector space.

 $Zb_1 + \ldots + Zb_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in Z\}$ is a rank-k length-n lattice.

 b_1, \ldots, b_k is a **basis** of this lattice.

Short vectors in la

Given b_1, b_2, \ldots, b_n what is shortest ve in $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_n$

44

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a k-dimensional vector space.

 $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1, \ldots, b_k is a **basis** of this lattice.

45

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a *k*-dimensional vector space.

 $Zb_1 + \ldots + Zb_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1,\ldots,b_k is a **basis** of this lattice.

45

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a *k*-dimensional vector space.

 $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1,\ldots,b_k is a **basis** of this lattice.

45

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a k-dimensional vector space.

 $Zb_1 + \ldots + Zb_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1, \ldots, b_k is a **basis** of this lattice.

45

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

What is shortest nonzero vector?

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a *k*-dimensional vector space.

 $Zb_1 + \ldots + Zb_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1, \ldots, b_k is a **basis** of this lattice.

45

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times

length of shortest nonzero vector.

Assume that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ are **R**-linearly independent, i.e., $\mathbf{R}b_1 + \ldots + \mathbf{R}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{R}\}$ is a k-dimensional vector space.

 $\mathbf{Z}b_1 + \ldots + \mathbf{Z}b_k =$ $\{r_1b_1 + \ldots + r_kb_k : r_1, \ldots, r_k \in \mathbf{Z}\}$ is a rank-k length-n lattice.

 b_1, \ldots, b_k is a **basis** of this lattice.

45

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$? 0.

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

mathematically

that $b_1, b_2, \ldots, b_k \in \mathbf{R}^n$ nearly independent,

 $+\ldots+\mathbf{R}b_{k}=$

 $\ldots + r_k b_k : r_1, \ldots, r_k \in \mathbf{R}$ mensional vector space.

 $\ldots + \mathbf{Z}b_k =$ $\ldots + r_k b_k : r_1, \ldots, r_k \in \mathbf{Z}$ k-k length-n lattice.

 b_k

s of this lattice.

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

45

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

Lattice v

Given pı Compute

tically

 $b_2, \ldots, b_k \in \mathbf{R}^n$ pendent,

 $b_k =$

 $r_1, \ldots, r_k \in \mathbf{R}$ vector space.

 $\{r_1, \ldots, r_k \in \mathbf{Z}\}$

attice.

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $\mathbb{Z}b_1 + \ldots + \mathbb{Z}b_k$?

0.

45

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

Lattice view of N7

Given public key A Compute A/3 = a

45

0.

 $\in \mathbf{R}^n$

 $r_k \in \mathbf{R}$ ace.

 $\mathsf{r}_k \in \mathsf{Z}$

Short vectors in lattices

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector. Fancier algorithms (e.g., BKZ) compute shorter vectors

at surprisingly high speed.

46

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

46

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

46

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$

by a few additions, subtractions.

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

46

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, ..., x^{n-1}$

by a few additions, subtractions.

d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

Given $b_1, b_2, \ldots, b_k \in \mathbb{Z}^n$, what is shortest vector in $Zb_1 + ... + Zb_k$?

0.

What is shortest nonzero vector?

LLL algorithm runs in poly time, computes a vector whose length is at most $2^{n/2}$ times length of shortest nonzero vector.

Fancier algorithms (e.g., BKZ) compute shorter vectors at surprisingly high speed.

46

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$

by a few additions, subtractions.

d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$

a is obtained from $q, qx, qx^2, \ldots, qx^{n-1},$ $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

47

by a few additions, subtractions.

ctors in lattices

 $b_2, \ldots, b_k \in \mathbf{Z}^n$, shortest vector $-\ldots+\mathbf{Z}b_k?$

- shortest nonzero vector?
- prithm runs in poly time, es a vector whose length st $2^{n/2}$ times
- f shortest nonzero vector.

algorithms (e.g., BKZ) e shorter vectors singly high speed.

Lattice view of NTRU

46

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$ by a few additions, subtractions. d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions. a is obtained from

 $q, qx, qx^2, \ldots, qx^{n-1},$ $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

(*a*, *d*) is (q, 0),(qx, 0), (qx^{n-1}) , (A/3, 1)(xA/3, x $(x^{n-1}A)$ by a few

ttices

 $\mathbf{p}_k \in \mathbf{Z}^n$,

ector

k?

onzero vector?

s in poly time,

whose length

nes

nonzero vector.

(e.g., BKZ)

ectors

n speed.

Lattice view of NTRU

46

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$ by a few additions, subtractions. d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions. a is obtained from $q, qx, qx^2, \ldots, qx^{n-1},$ $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

(*a*, *d*) is obtained (q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x), $(x^{n-1}A/3, x^{n-1})$ by a few additions

46

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$ by a few additions, subtractions. d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions. a is obtained from

 $q, qx, qx^2, \ldots, qx^{n-1},$ $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

(q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x),

47

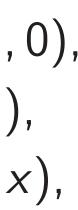
ctor?

zime, ngth

ector.

(Z)

(a, d) is obtained from



$(x^{n-1}A/3, x^{n-1})$ by a few additions, subtract

Lattice view of NTRU

Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$ by a few additions, subtractions.

d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

a is obtained from $q, qx, qx^{2}, \ldots, qx^{n-1},$ $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

(a, d) is obtained from (q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x), $(x^{n-1}A/3, x^{n-1})$ by a few additions, subtractions.

Lattice view of NTRU

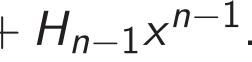
Given public key A = 3a/d. Compute A/3 = a/d.

d is obtained from $1, x, \ldots, x^{n-1}$ by a few additions, subtractions. d(A/3) is obtained from $A/3, xA/3, ..., x^{n-1}A/3$

by a few additions, subtractions.

a is obtained from $q, qx, qx^{2}, \ldots, qx^{n-1},$ $A/3, xA/3, ..., x^{n-1}A/3$ by a few additions, subtractions.

(a, d) is obtained from (q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x), $(x^{n-1}A/3, x^{n-1})$ by a few additions, subtractions. Write A/3 as $H_0 + H_1 x + \ldots + H_{n-1} x^{n-1}$.



view of NTRU

ublic key
$$A = 3a/d$$
.
e $A/3 = a/d$.

ained from x^{n-1}

^v additions, subtractions.

is obtained from

 $/3, ..., x^{n-1}A/3$

^v additions, subtractions.

ained from

 $x^{2}, \ldots, qx^{n-1},$ $/3, \ldots, x^{n-1}A/3$

^v additions, subtractions.

(a, d) is obtained from (q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x), $(x^{n-1}A/3, x^{n-1})$ by a few additions, subtractions. Write A/3 as $H_0 + H_1 x + \ldots + H_{n-1} x^{n-1}$.

 $(a_0, a_1, .$ is obtair (q, 0, . . . (0, q, . . . $(0, 0, \ldots, (H_0, H_1, H_1))$ (H_{n-1}, H_{n-1}) $(H_1, H_2,$ by a few

<u>rru</u>

A = 3a/d.

47

, subtractions.

d from $^{-1}A/3$

, subtractions.

 $^{n-1}, -{}^{1}A/3$

, subtractions.

(a, d) is obtained from (q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x), $(x^{n-1}A/3, x^{n-1})$ by a few additions, subtractions. Write A/3 as $H_0 + H_1 x + \ldots + H_{n-1} x^{n-1}$.

$(a_0, a_1, \ldots, a_{n-1}, a_{n-1})$ is obtained from $(q, 0, \ldots, 0, 0, 0, \ldots)$ $(0, q, \ldots, 0, 0, 0, \ldots)$ $(0, 0, \ldots, q, 0, 0, \ldots, (H_0, H_1, \ldots, H_{n-1}))$ $(H_1, H_2, \ldots, H_0, 0)$

48

by a few additions

ions.

47

ions.

ions.

(a, d) is obtained from
(q, 0),
(qx, 0),
:
(qxⁿ⁻¹, 0),
(A/3, 1),
(xA/3, x),
:
(xⁿ⁻¹A/3, xⁿ⁻¹)
by a few additions, subtractions
Write A/3 as

$$H_0 + H_1x + \ldots + H_{n-1}x^{n-1}$$
.

48

 $(a_0, a_1, ..., a_{n-1}, d_0, d_1, ...,$ is obtained from

 $(q, 0, \ldots, 0, 0, 0, \ldots, 0),$ $(0, q, \ldots, 0, 0, 0, \ldots, 0),$

 $(0, 0, \ldots, q, 0, 0, \ldots, 0),$ $(H_0, H_1, \ldots, H_{n-1}, 1, 0, \ldots, (H_{n-1}, H_0, \ldots, H_{n-2}, 0, 1, \ldots))$

 $(H_1, H_2, \ldots, H_0, 0, 0, \ldots, 1)$ by a few additions, subtract

(a, d) is obtained from (q, 0),(qx, 0), $(qx^{n-1}, 0),$ (A/3, 1),(xA/3, x), $(x^{n-1}A/3, x^{n-1})$ by a few additions, subtractions.

Write A/3 as $H_0 + H_1 x + \ldots + H_{n-1} x^{n-1}$.

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is obtained from $(q, 0, \ldots, 0, 0, 0, \ldots, 0),$ $(0, q, \ldots, 0, 0, 0, \ldots, 0),$ $(0, 0, \ldots, q, 0, 0, \ldots, 0),$ $(H_0, H_1, \ldots, H_{n-1}, 1, 0, \ldots, 0),$ $(H_{n-1}, H_0, \ldots, H_{n-2}, 0, 1, \ldots, 0),$ $(H_1, H_2, \ldots, H_0, 0, 0, \ldots, 1)$ by a few additions, subtractions.

48

obtained from

0),

(),

 $(3, x^{n-1})$

^v additions, subtractions.

/3 as $x+\ldots+H_{n-1}x^{n-1}$.

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots)$ is obtained from $(q, 0, \ldots, 0, 0, 0, \ldots, 0),$ $(0, q, \ldots, 0, 0, 0, \ldots, 0),$ $(0, 0, \ldots, q, 0, 0, \ldots, 0),$ $(H_0, H_1, \ldots, H_{n-1}, 1, 0, \ldots, 0),$ $(H_{n-1}, H_0, \ldots, H_{n-2}, 0, 1, \ldots, 0),$ $(H_1, H_2, \ldots, H_0, 0, 0, \ldots, 1)$ by a few additions, subtractions.

48

,
$$d_{n-1})$$

49

$(a_0, a_1, .$ is a surp in lattice $(q, 0, \ldots)$

from

48

, subtractions.

$$H_{n-1}x^{n-1}$$

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is obtained from $(q, 0, \ldots, 0, 0, 0, \ldots, 0),$ $(0, q, \ldots, 0, 0, 0, \ldots, 0),$ $(0, 0, \dots, q, 0, 0, \dots, 0),$ $(H_0, H_1, \dots, H_{n-1}, 1, 0, \dots, 0),$ $(H_{n-1}, H_0, \dots, H_{n-2}, 0, 1, \dots, 0),$ $(H_1, H_2, \ldots, H_0, 0, 0, \ldots, 1)$ by a few additions, subtractions.

$(a_0, a_1, \ldots, a_{n-1}, a_n)$ is a surprisingly sh in lattice generate $(q, 0, \ldots, 0, 0, 0, \ldots)$

$$(a_{0}, a_{1}, \dots, a_{n-1}, d_{0}, d_{1}, \dots, d_{n-1})$$
(is obtained from
$$(q, 0, \dots, 0, 0, 0, \dots, 0),$$

$$(0, q, \dots, 0, 0, 0, \dots, 0),$$

$$(0, 0, \dots, q, 0, 0, \dots, 0),$$

$$(H_{0}, H_{1}, \dots, H_{n-1}, 1, 0, \dots, 0),$$

$$(H_{n-1}, H_{0}, \dots, H_{n-2}, 0, 1, \dots, 0),$$

$$(H_{1}, H_{2}, \dots, H_{0}, 0, 0, \dots, 1)$$
by a few additions, subtractions.

ions.

48

1.

$(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, a_n)$'s a surprisingly short vector In lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

49

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

49

Attacker searches for short vector in this lattice using LLL etc.

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc. Attacker searches for short vector in this lattice using LLL etc. 1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a.

49

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc. Attacker searches for short vector in this lattice using LLL etc. 1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a. Exercise: Describe search for (b, c) as a problem of finding

49

- a vector close to a lattice.

..., a_{n-1} , d_0 , d_1 , ..., d_{n-1}) ed from

49

 $, 0, 0, 0, \ldots, 0),$ $, 0, 0, 0, \ldots, 0),$

$$, q, 0, 0, ..., 0),$$

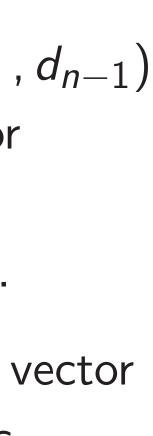
..., $H_{n-1}, 1, 0, ..., 0),$
 $H_0, ..., H_{n-2}, 0, 1, ..., 0),$

..., H_0 , 0, 0, ..., 1) ^v additions, subtractions. $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

Attacker searches for short vector in this lattice using LLL etc.

1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a.

Exercise: Describe search for (*b*, *c*) as a problem of finding a vector close to a lattice.



Quotien

- "Quotie
- is the st
- Alice ge for smal
- i.e., *dA*

49
$$d_0, d_1, \ldots, d_{n-1}$$

.,0), .,0),

```
.,0),
,1,0,...,0),
____,0,1,...,0),
```

, 0, . . . , 1) , subtractions. $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

Attacker searches for short vector in this lattice using LLL etc.

1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if *d* is chosen $10 \times$ larger than *a*.

Exercise: Describe search for (b, c) as a problem of finding a vector close to a lattice.

Quotient NTRU v

"Quotient NTRU" is the structure we

Alice generates A

for small random a

i.e., dA - 3a = 0

 $d_{n-1})$

49

0), .,0),

ions.

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

Attacker searches for short vector in this lattice using LLL etc.

1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a.

Exercise: Describe search for (*b*, *c*) as a problem of finding a vector close to a lattice.

50

Quotient NTRU vs. product

"Quotient NTRU" (new nar is the structure we've seen:

Alice generates A = 3a/d in

for small random *a*, *d*:

i.e., dA - 3a = 0 in R_a .

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

Attacker searches for short vector in this lattice using LLL etc.

1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a.

Exercise: Describe search for (b, c) as a problem of finding a vector close to a lattice.

50

51 Quotient NTRU vs. product NTRU

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_a for small random *a*, *d*: i.e., dA - 3a = 0 in R_a .

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

Attacker searches for short vector in this lattice using LLL etc.

1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a.

Exercise: Describe search for (b, c) as a problem of finding a vector close to a lattice.

50

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_a for small random *a*, *d*: i.e., dA - 3a = 0 in R_a .

Bob sends C = Ab + c in R_a . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Quotient NTRU vs. product NTRU

 $(a_0, a_1, \ldots, a_{n-1}, d_0, d_1, \ldots, d_{n-1})$ is a surprisingly short vector in lattice generated by $(q, 0, \ldots, 0, 0, 0, \ldots, 0)$ etc.

Attacker searches for short vector in this lattice using LLL etc.

1997 Coppersmith–Shamir balancing: e.g., set up lattice to contain (10a, d)if d is chosen $10 \times$ larger than a.

Exercise: Describe search for (b, c) as a problem of finding a vector close to a lattice.

50

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_a for small random *a*, *d*: i.e., dA - 3a = 0 in R_a .

Bob sends C = Ab + c in R_a . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Alice reconstructs 3ab + dc in R, using smallness of a, b, d, c. Alice computes dc in R_3 , deduces c, deduces b.

Quotient NTRU vs. product NTRU

..., a_{n-1} , d_0 , d_1 , ..., d_{n-1}) orisingly short vector e generated by , 0, 0, 0, ..., 0) etc. 50

searches for short vector strice using LLL etc.

ppersmith–Shamir g: e.g., set up lattice in (10*a, d*) nosen 10× larger than *a*.

: Describe search for

a problem of finding close to a lattice.

Quotient NTRU vs. product NTRU

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_q for small random a, d: i.e., dA - 3a = 0 in R_q .

Bob sends C = Ab + c in R_q . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Alice reconstructs 3ab + dc in R, using smallness of a, b, d, c. Alice computes dc in R_3 , deduces c, deduces b.

51

"Produc 2010 Ly

Everyone Alice get for smal $d_0, d_1, \ldots, d_{n-1})$ ort vector d by .,0) etc. for short vector g LLL etc. -Shamir

50

- t up lattice
- larger than a.
- e search for
- n of finding
- lattice.

Quotient NTRU vs. product NTRU

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_q for small random *a*, *d*: i.e., dA - 3a = 0 in R_a .

Bob sends C = Ab + c in R_q . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Alice reconstructs 3ab + dc in R, using smallness of a, b, d, c. Alice computes dc in R_3 , deduces *c*, deduces *b*.

"Product NTRU" 2010 Lyubashevsk Everyone knows ra Alice generates A

51

for small random .

$d_{n-1})^{50}$	<u>Quotient NTRU vs. product NTR</u>
	"Quotient NTRU" (new name) is the structure we've seen:
vector	Alice generates $A = 3a/d$ in R_q for small random a, d : i.e., $dA - 3a = 0$ in R_q .
an <i>a</i> .	Bob sends $C = Ab + c$ in R_q . Alice computes dC in R_q , i.e., $3ab + dc$ in R_q .
r	Alice reconstructs $3ab + dc$ in R , using smallness of a , b , d , c . Alice computes dc in R_3 , deduces c , deduces b .

51 U vs. product NTRU RU" (new name) e we've seen: s A = 3a/d in R_q om *a, d* :

"Product NTRU" (new nam 2010 Lyubashevsky-Peikert-

Everyone knows random $G \in$

Alice generates A = aG + d

for small random *a*, *d*.

51 Quotient NTRU vs. product NTRU

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_a for small random *a*, *d*: i.e., dA - 3a = 0 in R_q .

Bob sends C = Ab + c in R_q . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Alice reconstructs 3ab + dc in R, using smallness of a, b, d, c. Alice computes dc in R_3 , deduces c, deduces b.

"Product NTRU" (new name),

Everyone knows random $G \in R_q$. for small random *a*, *d*.

2010 Lyubashevsky–Peikert–Regev: Alice generates A = aG + d in R_q

51 Quotient NTRU vs. product NTRU

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_a for small random *a*, *d*: i.e., dA - 3a = 0 in R_q .

Bob sends C = Ab + c in R_q . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Alice reconstructs 3ab + dc in R, using smallness of a, b, d, c. Alice computes dc in R_3 , deduces *c*, deduces *b*.

"Product NTRU" (new name), Everyone knows random $G \in R_q$. Alice generates A = aG + d in R_q for small random *a*, *d*. Bob sends B = Gb + e in R_a and C = m + Ab + c in R_a where b, c, e are small and

2010 Lyubashevsky–Peikert–Regev:

- each coefficient of m is 0 or q/2.

51 Quotient NTRU vs. product NTRU

"Quotient NTRU" (new name) is the structure we've seen:

Alice generates A = 3a/d in R_q for small random *a*, *d*: i.e., dA - 3a = 0 in R_q .

Bob sends C = Ab + c in R_q . Alice computes dC in R_q , i.e., 3ab + dc in R_q .

Alice reconstructs 3ab + dc in R, using smallness of a, b, d, c. Alice computes dc in R_3 , deduces c, deduces b.

"Product NTRU" (new name), Everyone knows random $G \in R_q$. Alice generates A = aG + d in R_q for small random *a*, *d*. Bob sends B = Gb + e in R_a and C = m + Ab + c in R_q where *b*, *c*, *e* are small and each coefficient of *m* is 0 or q/2. Alice computes C - aB in R_a , i.e., m + db + c - ae in R_a . Alice reconstructs m, using smallness of d, b, c, a, e.

2010 Lyubashevsky–Peikert–Regev: