Google

Frodo: Take off the Ring!
Practical Quantum-Secure Key Encapsulation from Lattices

Ananth Raghunathan

FSTTCS Lattice Algorithms and Cryptography Workshop (Dec 2017)

Acknowledgments

Erdem Alkim*

Joppe Bos

Léo Ducas

Craig Costello †

Karen Easterbrook

Brian LaMacchia

Patrick Longa*

Michael Naehrig

Ilya Mironov

Valeria Nikolaenko

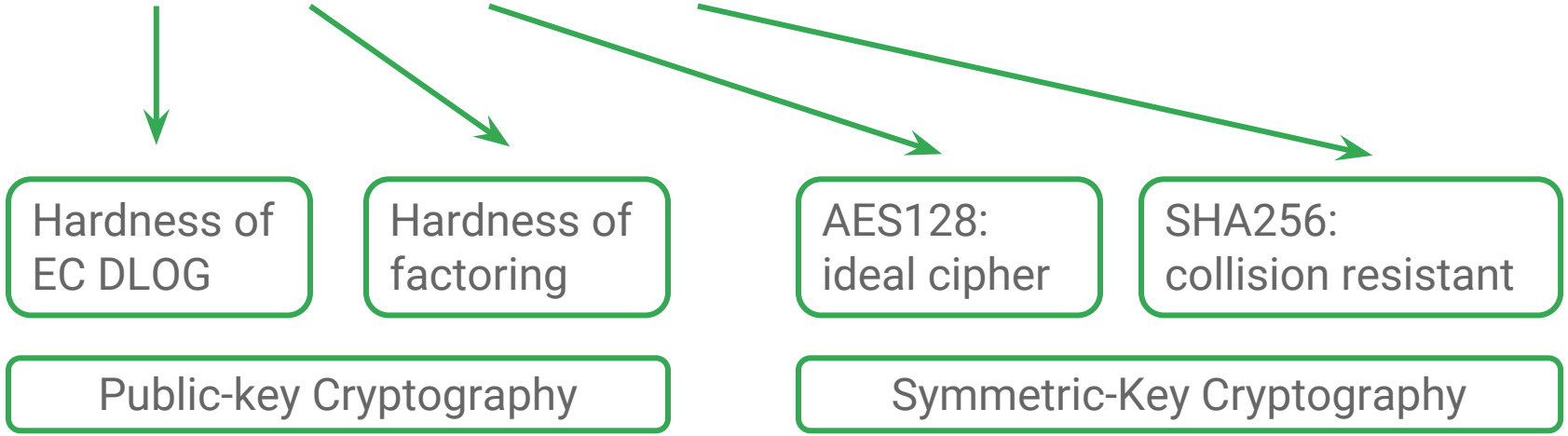
Chris Peikert*

Douglas Stebila

* – NIST submission, † – CCS paper

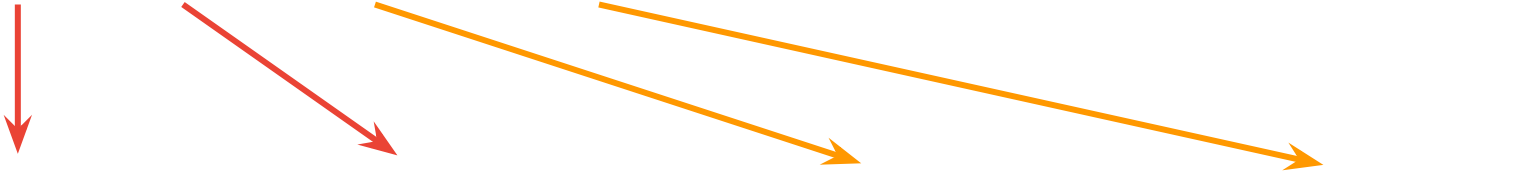
Contemporary Cryptography

`TLS-ECDHE-RSA-AES128-GCM-SHA256`



Contemporary Cryptography (with a Quantum Computer)

TLS-~~ECDHE~~-~~RSA~~-AES128-GCM-SHA256



Hardness of EC DLOG

Hardness of factoring

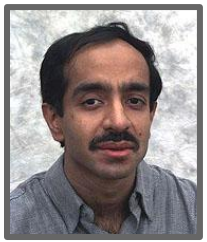
AES128: ideal cipher

SHA256: collision resistant

Public-key Cryptography

Symmetric-Key Cryptography

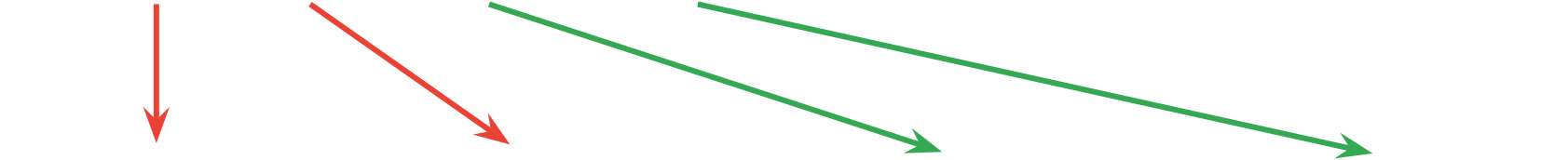
Shor's algorithm (1994)



Grover's algorithm (1996)

Contemporary Cryptography (with a Quantum Computer)

TLS-~~ECDHE~~-~~RSA~~-AES256-GCM-SHA384 ("Double" Key Sizes!)



Hardness of EC DLOG

Hardness of factoring

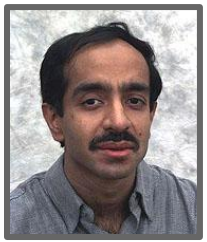
AES256: ideal cipher

SHA384: collision resistant

Public-key Cryptography

Symmetric-Key Cryptography

Shor's algorithm (1994)



Grover's algorithm (1996)

Practical Quantum Computers(?)

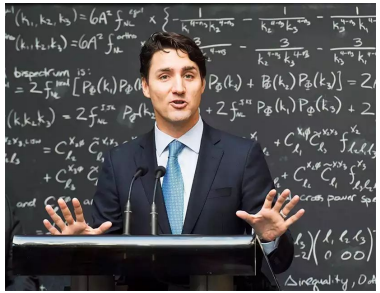
Major investments in industry

Microsoft intel NATIONAL SECURITY AGENCY UNITED STATES OF AMERICA Google D:wave The Quantum Computing Company™

Major research in academia

http://www.quantiki.org/groups

Major interest from politicians



Justin Trudeau, Canadian PM, Apr. 2016

Quantum Computing Getting More Practical

Intelligent Machines

TECHNOLOGY

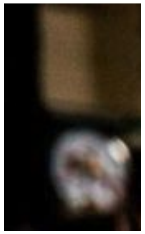
IB

W

Q

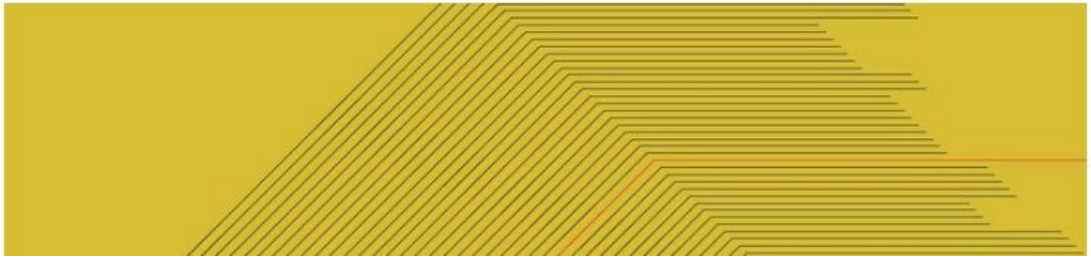
Yale

By CADE ME



WILL HURD SECURITY 12.07.17 08:00 AM

QUANTUM COMPUTING IS THE NEXT BIG SECURITY RISK



Computer

“I estimate a 1-in-7 chance of breaking RSA-2048 by 2026, and a 1-in-2 chance by 2031.”

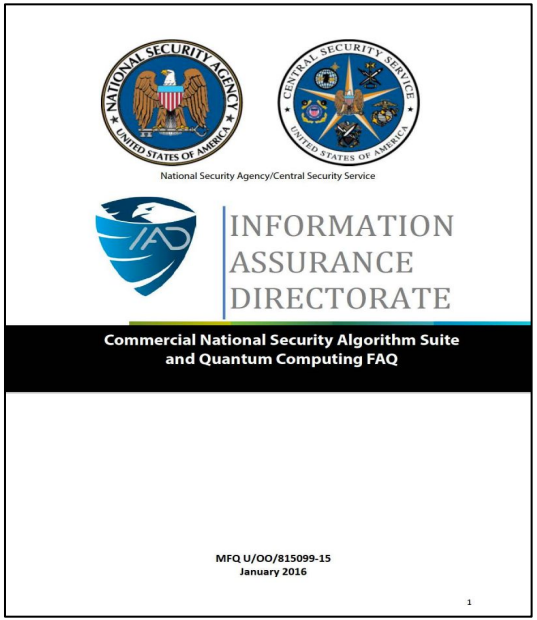
—Michele Mosca, Nov. 2015
(eprint/2015/1075)

“The crypt-apocalypse is nigh!”

—Kaspersky Security Bulletin (Dec 2015)

https://securelist.com/files/2015/11/KSB_2016_Predictions_FINAL.pdf

What kicked things off?


National Security Agency/Central Security Service

INFORMATION ASSURANCE DIRECTORATE

Commercial National Security Algorithm Suite and Quantum Computing FAQ

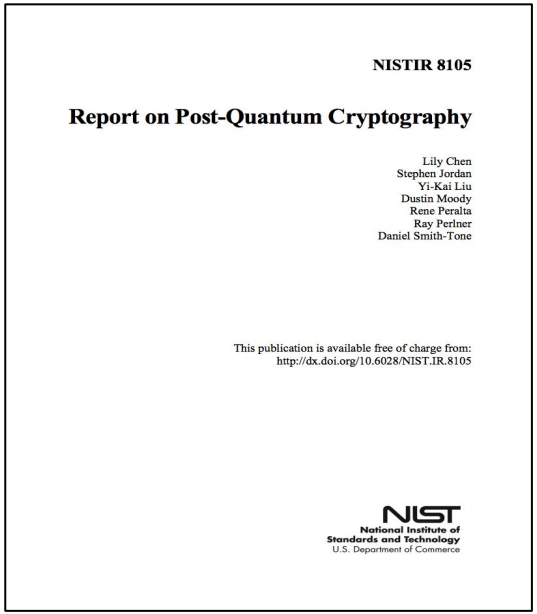
MFQ U/OO/815099-15
January 2016

1

Jan. 2016

"IAD will initiate a transition to quantum resistant algorithms in the not too distant future."

—NSA Information Assurance Directorate, Aug. 2015


NISTIR 8105

Report on Post-Quantum Cryptography

Lily Chen
Stephen Jordan
Yi-Kai Liu
Dustin Moody
Rene Peralta
Ray Perlner
Daniel Smith-Tone

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.IR.8105

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Apr. 2016

NIST Timeline

September 2016	Feedback on call for proposals
Fall 2016	Formal call for proposals
Nov. 30, 2017	Deadline for submissions
Early 2018	Workshop—submitter's presentations
3–5 years	Analysis phase
2 years	Draft standards ready

Post-Quantum or Quantum-Safe Crypto

"No known exponential quantum speedup"

Lattices	24	4	**28**
Codes	19	5	**24**
Multivariate	6	7	**13**
Hash	-	4	**4**

Others (Braids, Chebyshev, Finite Automata, Hypercomplex No.s, Isogeny, Random Walk)	7	2	**9**
RSA	1	1	**2**

Source: (NIST) https://twitter.com/DemocraticLuntz/status/937702005631586304

What are the challenges?

- Design better post-quantum key exchange and signature schemes
- Improve classical and quantum attacks
- Pick parameter sizes
- Develop fast, secure implementations
- Integrate them into the existing infrastructure

This Talk: Key-Exchange

*Premise: large-scale quantum computers don't exist right now, but we want to protect **today's communication** against **tomorrow's adversary***

TLS traffic, messages encrypted and saved **today** vulnerable to attacks **decades from now**.

Signatures can still be done with traditional primitives; require **online** quantum computer to cause havoc.

JAMES BAMFORD 03.15.12 7:24 PM

THE NSA IS BUILDING THE COUNTRY'S BIGGEST SPY CENTER (WATCH WHAT YOU SAY)

Courtesy Wired

Motivation

Lattice Background

Frodo Key Exchange (ACM CCS 2016)

FrodoKEM: our NIST proposal

Motivation

Lattice Background

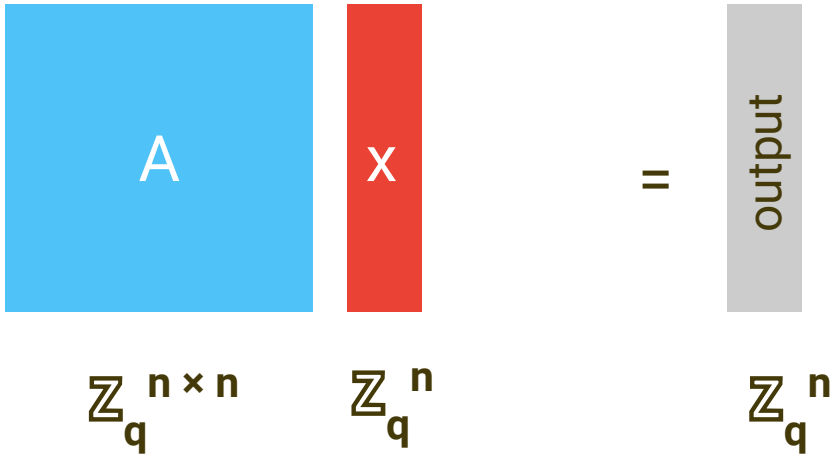
Frodo Key Exchange (ACM CCS 2016)

FrodoKEM: our NIST proposal

Learning with Errors

Random matrix A, secret x and output

Find x?

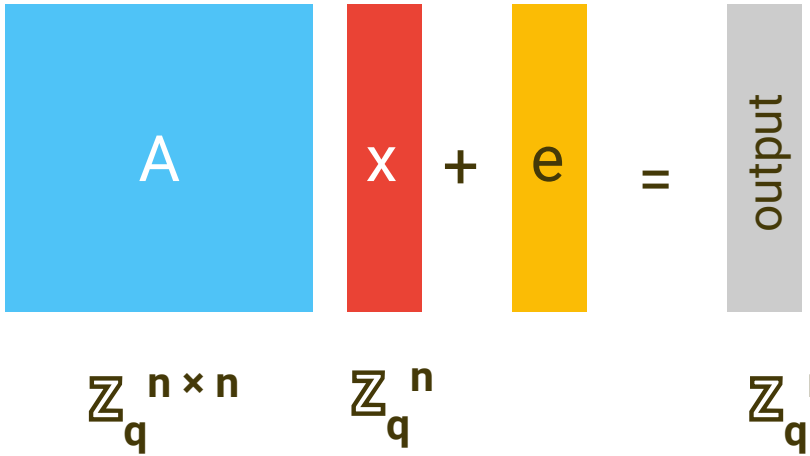


✓

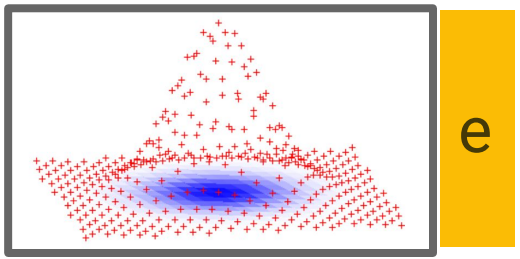
Easily solved using Gaussian elimination (Linear Algebra 101)

Learning with Errors [Regev '05]

Random matrix A, secret x, **small noise e**, and output. **Find x?**



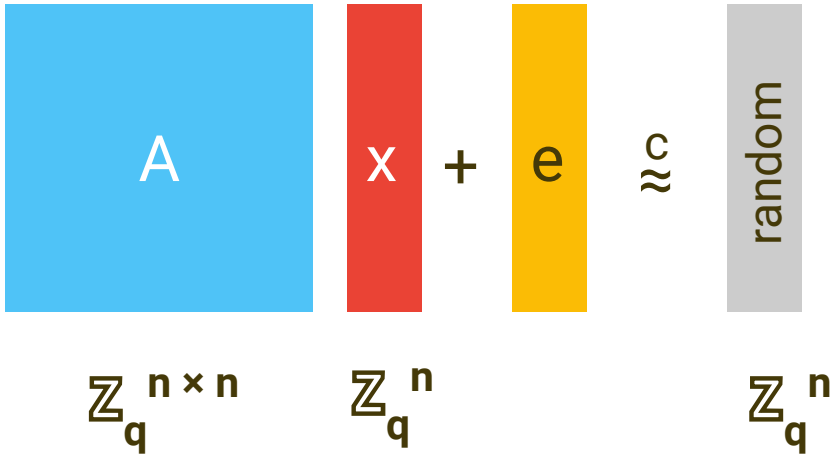
Search Learning with Errors problem



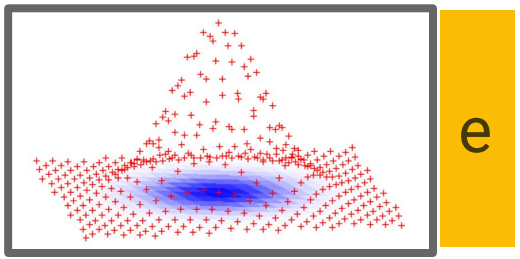
Learning with Errors [Regev '05]

Random matrix A, secret x, **small noise e**.

Distinguish (A, Ax+e) from (A, random)?



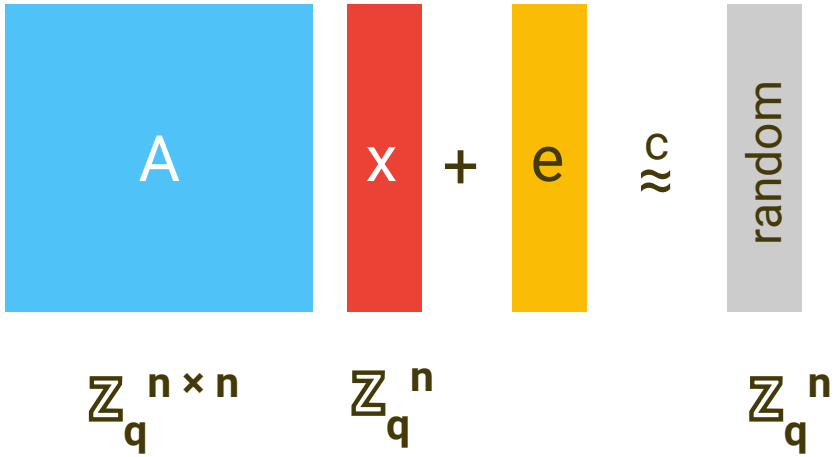
Decision Learning with Errors problem



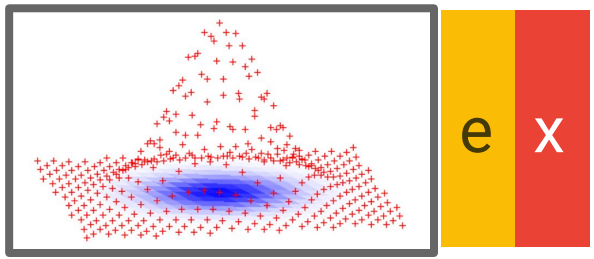
Learning with Errors with Short Secrets [Regev '05]

Random matrix A, **small** secret x, **small noise e**.

Distinguish (A, Ax+e) from (A, random)?



Decision Learning with Errors (w/ short secrets) problem



Other Variants

What if A has additional structure?
E.g., A is ***cyclic*** (each row is a shift of the row above), we get **Ring Learning with Errors [LPR '10]**
Closely related to: NTRU, M-LWE, NTRUPrime, …

Better key-sizes, faster (but complicated) arithmetic via FFT, …

Carefully instantiate **ring and noise**
Provably weak ring+noise instances **[ELOS '15, CLS '15, …]**
Quantum (sub-exponential) approximations for Ideal-SVP in some rings **[CGS '14, …]** (attacks don't affect RLWE at the present)

Frodo: Take off the Ring!
Practical, Quantum-Secure Key Exchange from LWE

Joppe Bos
NXP Semiconductors
joppe.bos@nxp.com

Craig Costello
Microsoft Research
craigco@microsoft.com

Léo Ducas
CWI
l.ducas@cwi.nl

Ilya Mironov
Google Inc.
mironov@google.com

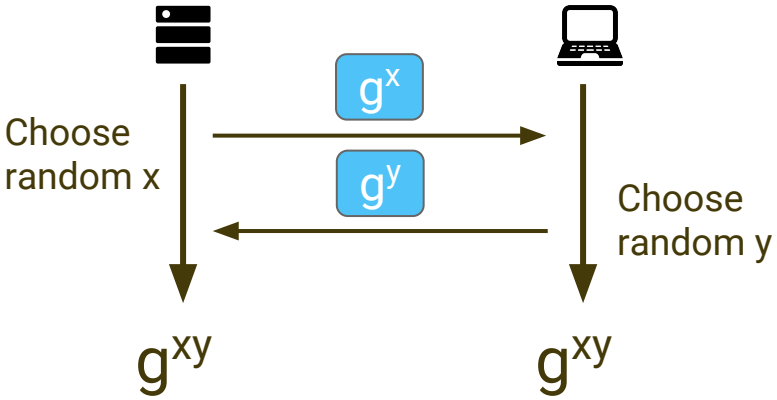
Michael Naehrig
Microsoft Research
mnaehrig@microsoft.com

Valeria Nikolaenko*
Stanford University
valerini@stanford.edu

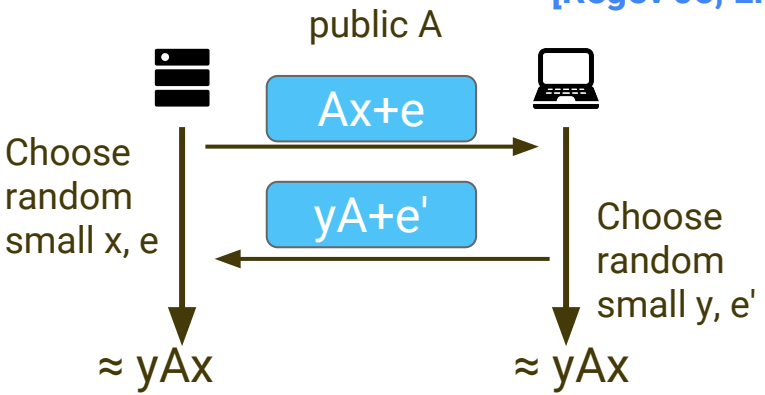
Ananth Raghunathan
Google Inc.
pseudorandom@google.com

Douglas Stebila
McMaster University
stebilad@mcmaster.ca

FrodoCCS: Key-Exchange



[Regev'05, LP'11]

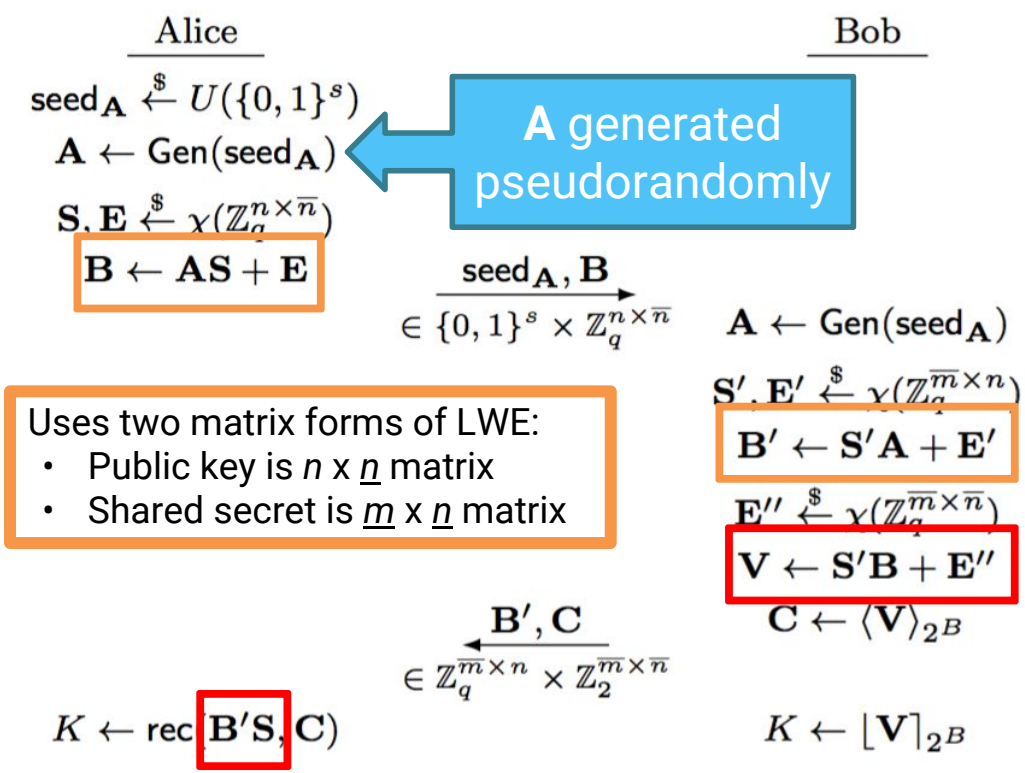


Short secrets to enable “close enough” values to be agreed upon

Key exchange now has a **failure probability**

Agreement between values—reconciliation!

FrodoCCS: Key-Exchange



Uses two matrix forms of LWE:

- Public key is *n* x *n* matrix
- Shared secret is *m* x *n* matrix

Assume same as A being generated uniformly at random

Decision Learning with Errors (w/ short secrets) problem

Approximate shared secret + reconciliation

FrodoCCS: Key-Exchange

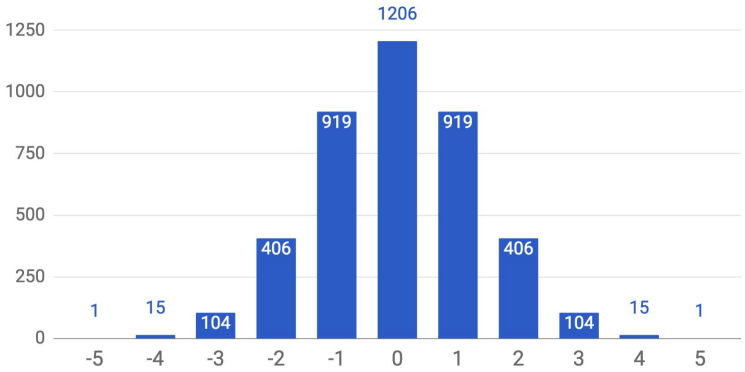
Rounding

We extract 4 bits from each entry in the shared secret
(Previous work only did 1-bit)

Parameter sizes, rounding, and error distribution all found via search scripts.

Error distribution through Rényi divergence

Fixed table over [-5, 5]
Close to discrete Gaussian
(R.D. of ~1.000301)
12 bits of randomness to sample



FrodoCCS: Parameters

“Recommended”

156-bit classical security,
142-bit quantum security,
112-bit plausible lower bound

- $n = 752, m = 8, q = 2^{15}$
- χ = approximation to rounded Gaussian with 11 elements
- Failure: $2^{-36.5}$
- Total communication: 22.6 KiB

“Paranoid”

191-bit classical security,
177-bit quantum security,
161-bit plausible lower bound

- $n = 864, m = 8, q = 2^{15}$
- χ = approximation to rounded Gaussian with 13 elements
- Failure: $2^{-35.8}$
- Total communication: 25.9 KiB

DEPRECATED!

FrodoCCS: Stand-alone benchmarks

DEPRECATED!

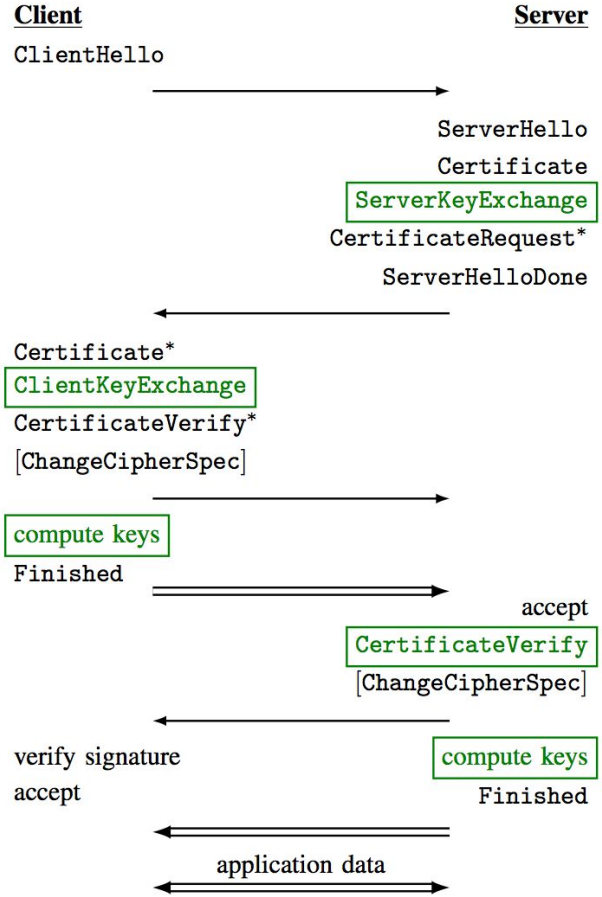
Scheme	Alice0 (ms)	Bob (ms)	Alice1 (ms)	Communication A→B	Communication B→A	Claimed security classical	Claimed security quantum
RSA 3072-bit	—	0.09	4.49	387	384	128	—
ECDH `nistp256`	0.366	0.698	0.331	32	32	128	—
BCNS	1.01	1.59	0.174	4,096	4,224	163	76
NewHope	0.112	0.164	0.034	1,824	2,048	229	206
NTRU `EES743EP1`	2.00	0.281	0.148	1,027	1,022	256	128
SIDH	135	464	301	564	564	192	128
Frodo Recomm.	**1.13**	**1.34**	**0.13**	**11,377**	**11,296**	**156**	**142**
Frodo Paranoid	1.25	1.64	0.15	13,057	12,976	191	174

FrodoCCS: Integration into TLS 1.2

New ciphersuite

`TLS-KEX-SIG-AES256-GCM-SHA384`

- SIG = RSA or ECDSA signatures for authentication
- KEX = Post-quantum key exchange
- AES-256 in GCM for authenticated encryption
- SHA-384 for HMAC-KDF



FrodoCCS: Security in TLS1.2

Model

Authenticated and Confidential Channel Establishment (ACCE) [JKSS '12]

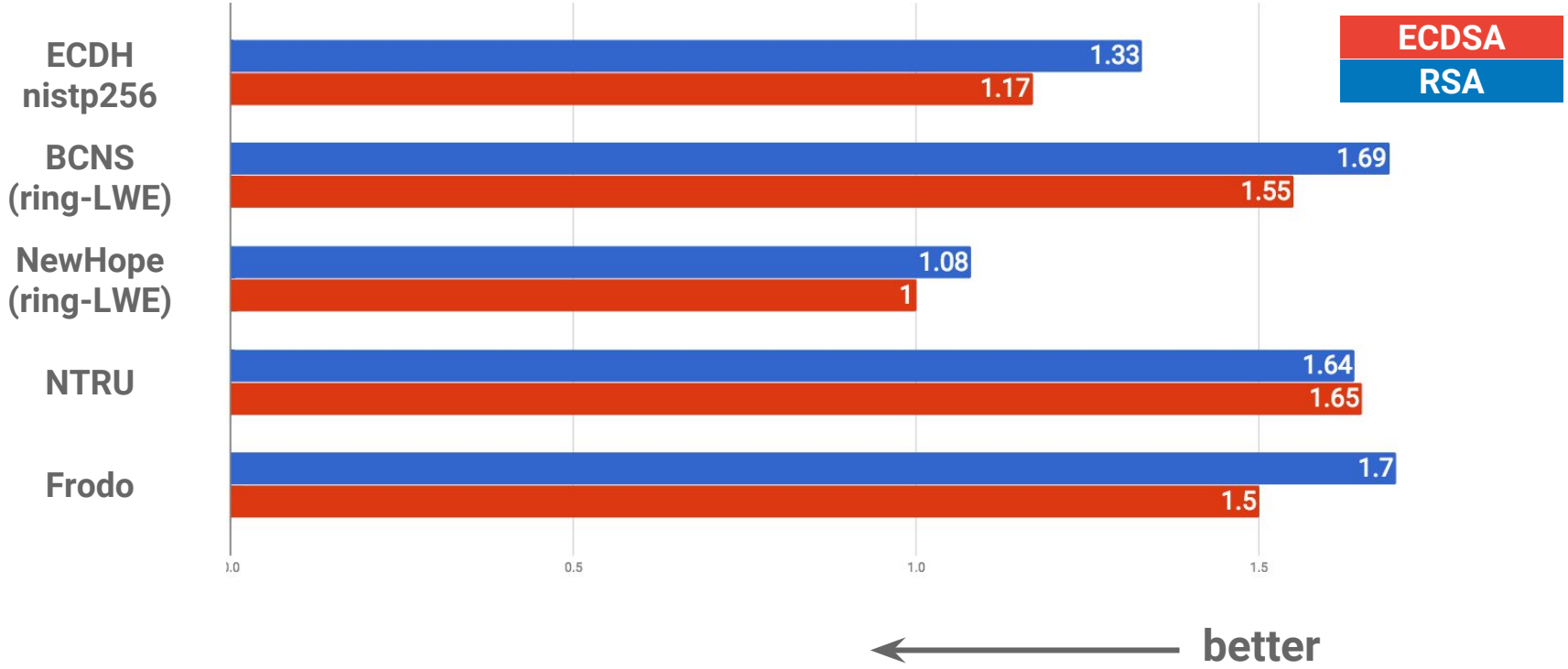
Theorem

Signed LWE/ring-LWE ciphersuite is ACCE-secure if underlying primitives (signatures, LWE/ring-LWE, authenticated encryption) are secure

Technical details—need to move server's signature to end of TLS handshake because oracle-DH assumptions don't hold for ring-LWE or use an IND-CCA KEM for key exchange

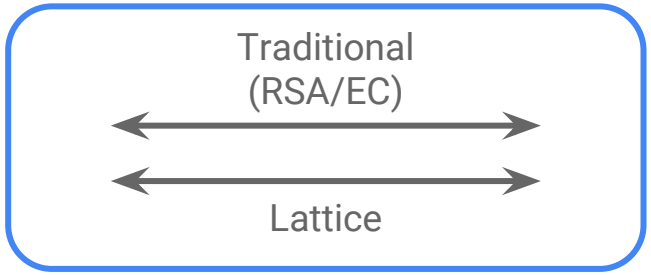
NIST submission (next part of the talk)

FrodoCCS: TLS1.2 Handshake Performance (v. NewHope-ECDSA)



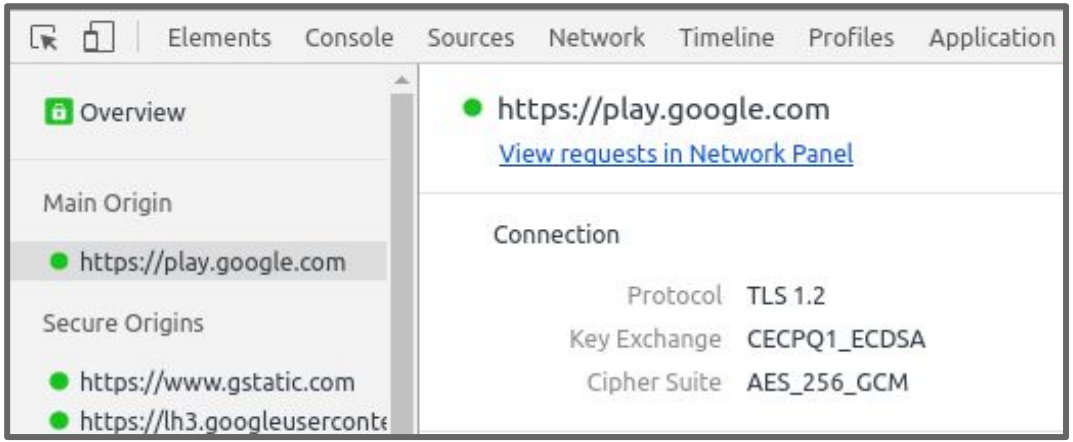
FrodoCCS: Hybrid ciphersuites

Hybrid Mode

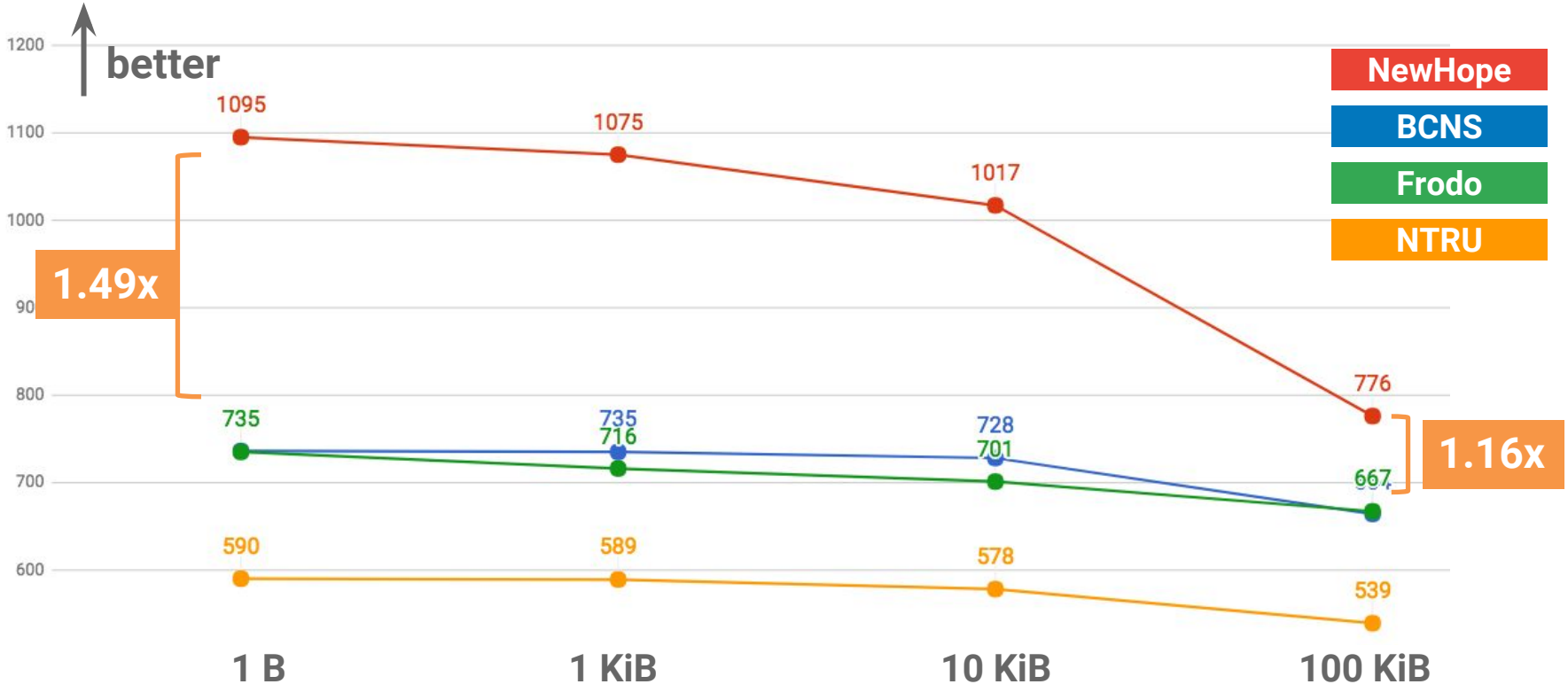


Session key is secure if ***either*** problem is hard

If post-quantum choices fail ***catastrophically*** we still have some baseline protection



FrodoCCS: TLS1.2 Throughput Performance (hybrid ECDHE/ECDSA)



Motivation

Lattice Background

Frodo Key Exchange (ACM CCS 2016)

FrodoKEM: our NIST proposal

FrodoKEM: our NIST proposal

- Generic, algebraically unstructured lattices
- Parameters from worst-case reductions ***and*** conservative cryptanalysis
  - “Wide enough” Gaussians with worst-case reductions
  - core-SVP hardness-based cryptanalysis estimates (NewHope, Frodo, CRYSTALS—Kyber, etc.)
- Simple design
  - Free modular arithmetic ($q = 2^{16}$)
  - Simple Gaussian sampling
  - Parallelizable matrix-vector operations
  - Key encapsulation without reconciliation
  - Simple code

FrodoKEM: our NIST proposal (contd.)

- Simple design
  - Free modular arithmetic ($q = 2^{16}$)
  - Simple Gaussian sampling
  - Parallelizable matrix-vector operations
  - Key encapsulation without reconciliation
  - Simple code
- Flexible, fine-grained choice of parameters (might support hom. enc.)
- Dynamically generated **A** to defend against all-for-the-price-of-one attacks (**AES** and **cSHAKE** variants)

FrodoCCS vs. FrodoKEM

FrodoCCS

Generic Lattices, Core-SVP hardness,
Free modular arithmetic,
Simple Gaussian sampling,
Dynamically generated **A** ✓

"Wide enough" Gaussians ✘

Key exchange with reconciliation

CPA-secure, small error rate

Security reduction outline ✘

FrodoKEM

Generic Lattices, Core-SVP hardness,
Free modular arithmetic,
Simple Gaussian sampling,
Dynamically generated **A** ✓

"Wide enough" Gaussians ✓

Key encapsulation

CCA-secure, ***negligible*** error rate

Security reduction outline ✓

FrodoKEM vs. FrodoCCS

Key exchange with reconciliation

CPA-secure, small error rate

Key encapsulation

CCA-secure, ***negligible*** error rate

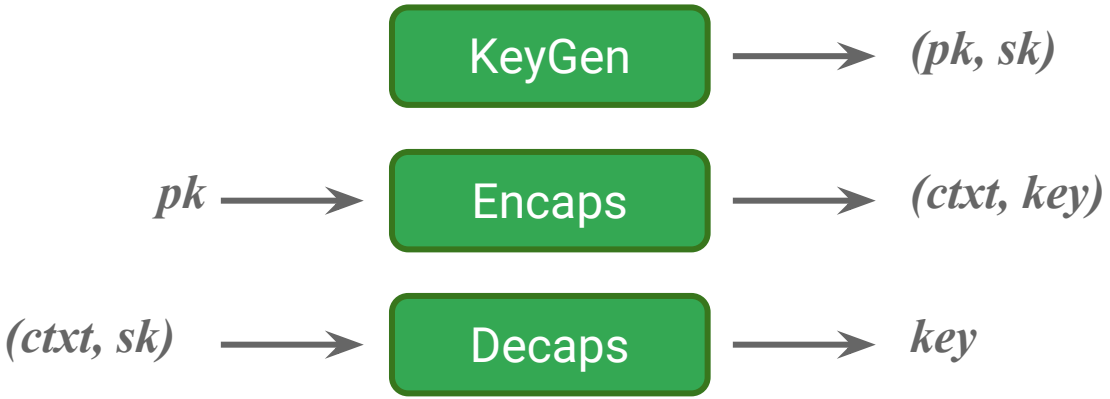
FrodoKEM vs. FrodoCCS

Key exchange with reconciliation

CPA-secure, small error rate

Key encapsulation

CCA-secure, ***negligible*** error rate



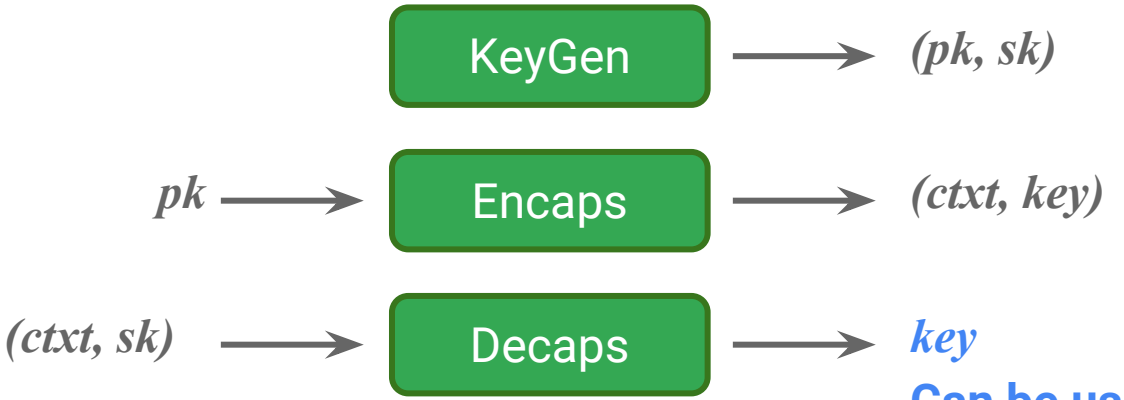
FrodoKEM vs. FrodoCCS

Key exchange with reconciliation

CPA-secure, small error rate

Key encapsulation

CCA-secure, ***negligible*** error rate



Can be used for symm. encryption!

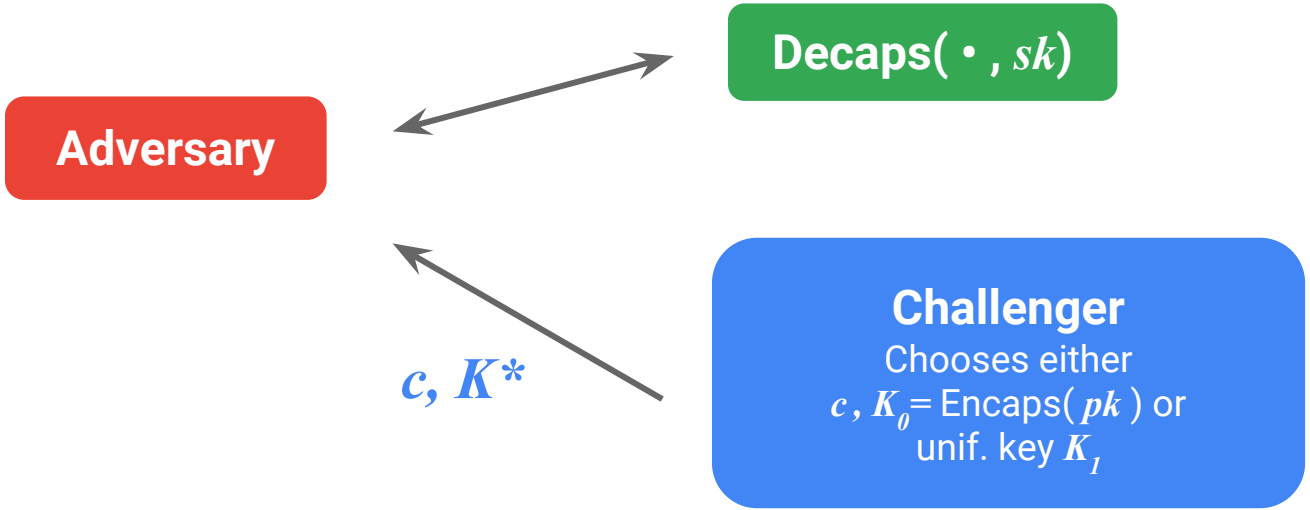
FrodoKEM vs. FrodoCCS

Key exchange with reconciliation

CPA-secure, small error rate

Key encapsulation

CCA-secure, ***negligible*** error rate



FrodoKEM: Outline

IND-CPA secure FrodoPKE

FrodoPKE.KeyGen

FrodoPKE.Enc

FrodoPKE.Dec

Algorithm 9 FrodoPKE.KeyGen.

Input: None.

Output: Key pair $(pk, sk) \in (\{0,1\}^{\mathsf{len_A}} \times \mathbb{Z}_q^{n \times \overline{n}}) \times \mathbb{Z}_q^{n \times \overline{n}}$.

1: Choose a uniformly random seed $\mathbf{seed_A} \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len_A}})$
2: Generate the matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ via $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathbf{seed_A})$
3: Choose a uniformly random seed $\mathbf{seed_E} \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len_E}})$
4: Sample error matrix $\mathbf{S} \leftarrow \mathsf{Frodo.SampleMatrix}(\mathbf{seed_E}, n, \overline{n}, T_\chi, 1)$
5: Sample error matrix $\mathbf{E} \leftarrow \mathsf{Frodo.SampleMatrix}(\mathbf{seed_E}, n, \overline{n}, T_\chi, 2)$
6: Compute $\mathbf{B} = \mathbf{AS} + \mathbf{E}$
7: **return** public key $pk \leftarrow (\mathbf{seed_A}, \mathbf{B})$ and secret key $sk \leftarrow \mathbf{S}$

FrodoKEM: Outline

IND-CPA secure FrodoPKE

FrodoPKE.KeyGen

FrodoPKE.Enc

FrodoPKE.Dec

Algorithm 10 FrodoPKE.Enc.

Input: Message $\mu \in \mathcal{M}$ and public key $pk = (\mathsf{seed}_{\mathbf{A}}, \mathbf{B}) \in \{0,1\}^{\mathsf{len}_{\mathbf{A}}} \times \mathbb{Z}_q^{n \times \overline{n}}$.

Output: Ciphertext $c = (\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{Z}_q^{\overline{m} \times n} \times \mathbb{Z}_q^{\overline{m} \times \overline{n}}$.

1: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_{\mathbf{A}})$
2: Choose a uniformly random seed $\mathsf{seed}_{\mathbf{E}} \leftarrow_{\$} U(\{0,1\}^{\mathsf{len}_{\mathbf{E}}})$
3: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}_{\mathbf{E}}, \overline{m}, n, T_\chi, 4)$
4: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}_{\mathbf{E}}, \overline{m}, n, T_\chi, 5)$
5: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}_{\mathbf{E}}, \overline{m}, \overline{n}, T_\chi, 6)$
6: Compute $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$ and $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
7: **return** ciphertext $c \leftarrow (\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + \mathsf{Frodo.Encode}(\mu))$

FrodoKEM: Outline

IND-CPA secure FrodoPKE

- FrodoPKE.KeyGen
- FrodoPKE.Enc
- FrodoPKE.Dec

Encode messages in B-bits chunks into the MSB

FrodoKEM: Outline

IND-CPA secure FrodoPKE

FrodoPKE.KeyGen

FrodoPKE.Enc

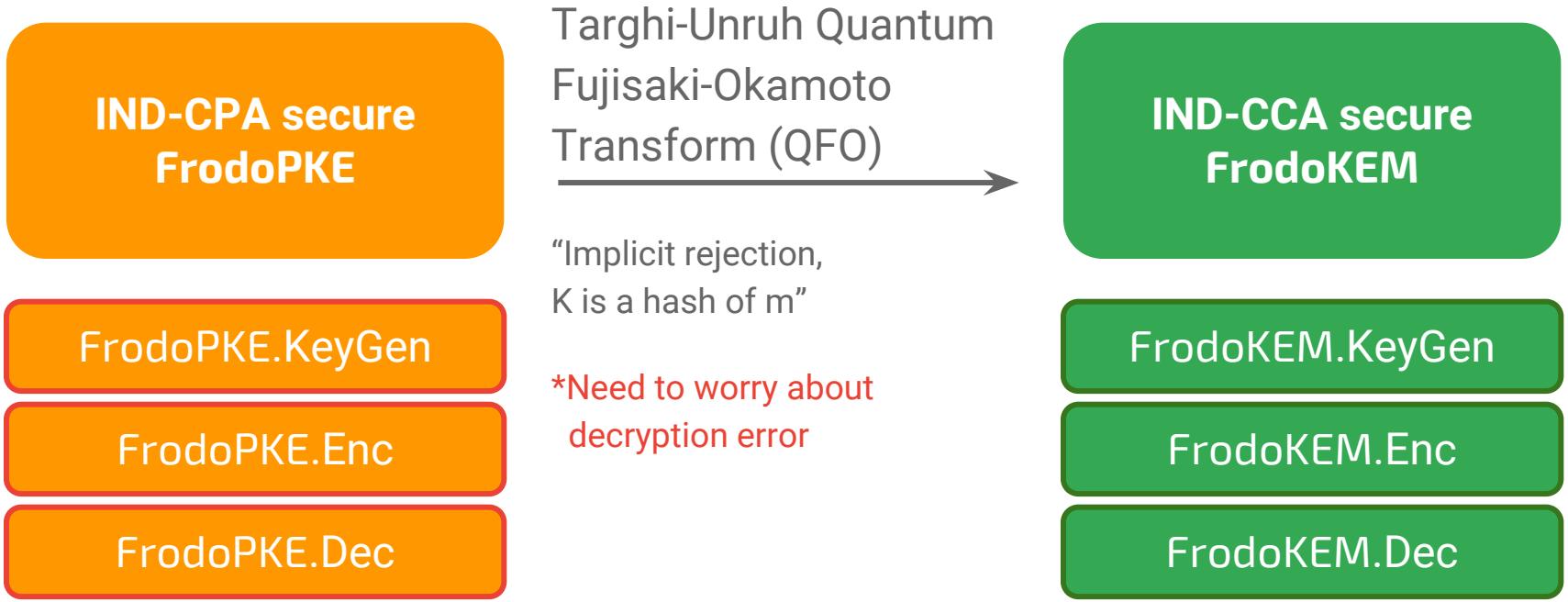
FrodoPKE.Dec

Algorithm 11 FrodoPKE.Dec.

Input: Ciphertext $c = (\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{Z}_q^{\overline{m} \times n} \times \mathbb{Z}_q^{\overline{m} \times \overline{n}}$ and secret key $sk = \mathbf{S} \in \mathbb{Z}_q^{n \times \overline{n}}$.
Output: Decrypted message $\mu' \in \mathcal{M}$.

1: Compute $\mathbf{M} = \mathbf{C}_2 - \mathbf{C}_1\mathbf{S}$
2: **return** message $\mu' \leftarrow$ Frodo.Decode($\mathbf{M}$)

FrodoKEM: Outline



FrodoKEM: Outline

Algorithm 12 FrodoKEM.KeyGen.

Input: None.

Output: Key pair (pk, sk') with $pk \in \{0,1\}^{\mathsf{len_A}+D\cdot n\cdot\overline{n}}$, $sk' \in \{0,1\}^{\mathsf{len_S}+\mathsf{len_A}+D\cdot}$

1: Choose uniformly random seeds $\mathbf{s}\|\mathsf{seed_E}\|\mathbf{z} \leftarrow_{\$} U(\{0,1\}^{\mathsf{len_S}+\mathsf{len_E}+\mathsf{len_z}})$
2: Generate pseudorandom seed $\mathsf{seed_A} \leftarrow H(\mathbf{z})$
3: Generate the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ via $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed_A})$
4: Sample error matrix $\mathbf{S} \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed_E}, n, \overline{n}, T_\chi, 1)$
5: Sample error matrix $\mathbf{E} \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed_E}, n, \overline{n}, T_\chi, 2)$
6: Compute $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
7: Compute $\mathbf{b} \leftarrow \mathsf{Frodo.Pack}(\mathbf{B})$
8: **return** public key $pk \leftarrow \mathsf{seed_A}\|\mathbf{b}$ and secret key $sk' \leftarrow (\mathbf{s}\|\mathsf{seed_A}\|\mathbf{b}, \mathbf{S})$

IND-CCA secure FrodoKEM

FrodoKEM.KeyGen

FrodoKEM.Enc

FrodoKEM.Dec

FrodoKEM: Outline

Algorithm 13 FrodoKEM.Encaps.

Input: Public key $pk = \mathsf{seed}_{\mathbf{A}} \| \mathbf{b} \in \{0,1\}^{\mathsf{len}_{\mathbf{A}} + D \cdot n \cdot \overline{n}}$.

Output: Ciphertext $\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{d} \in \{0,1\}^{(\overline{m} \cdot n + \overline{m} \cdot \overline{n}) D + \mathsf{len}_{\mathbf{d}}}$ and shared secret $\mathbf{ss} \in$

1: Choose a uniformly random key $\mu \leftarrow\!\!\$\ U(\{0,1\}^{\mathsf{len}_\mu})$
2: Generate pseudorandom values $\mathsf{seed}_{\mathbf{E}} \| \mathbf{k} \| \mathbf{d} \leftarrow G(pk \| \mu)$
3: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}_{\mathbf{E}}, \overline{m}, n, T_\chi, 4)$
4: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}_{\mathbf{E}}, \overline{m}, n, T_\chi, 5)$
5: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_{\mathbf{A}})$
6: Compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
7: Compute $\mathbf{c}_1 \leftarrow \mathsf{Frodo.Pack}(\mathbf{B}')$
8: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}_{\mathbf{E}}, \overline{m}, \overline{n}, T_\chi, 6)$
9: Compute $\mathbf{B} \leftarrow \mathsf{Frodo.Unpack}(\mathbf{b}, n, \overline{n})$
10: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
11: Compute $\mathbf{C} \leftarrow \mathbf{V} + \mathsf{Frodo.Encode}(\mu)$
12: Compute $\mathbf{c}_2 \leftarrow \mathsf{Frodo.Pack}(\mathbf{C})$
13: Compute $\mathbf{ss} \leftarrow F(\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{k} \| \mathbf{d})$
14: **return** ciphertext $\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{d}$ and shared secret $\mathbf{ss}$

IND-CCA secure FrodoKEM

FrodoKEM.KeyGen

FrodoKEM.Enc

FrodoKEM.Dec

FrodoKEM: Outline

Algorithm 14 FrodoKEM.Decaps.

Input: Ciphertext $\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{d} \in \{0,1\}^{(\overline{m} \cdot n + \overline{m} \cdot \overline{n}) D + \mathsf{len_d}}$, secret key $sk' = (\mathbf{s} \| \mathsf{seed}_\mathbf{A} \| \mathbf{b}, \mathbf{S}) \in \{0, \mathbb{Z}_q^{n \times \overline{n}}$.

Output: Shared secret $\mathbf{ss} \in \{0,1\}^{\mathsf{len_{ss}}}$.

1: $\mathbf{B}' \leftarrow \mathsf{Frodo.Unpack}(\mathbf{c}_1)$
2: $\mathbf{C} \leftarrow \mathsf{Frodo.Unpack}(\mathbf{c}_2)$
3: Compute $\mathbf{M} \leftarrow \mathbf{C} - \mathbf{B}'\mathbf{S}$
4: Compute $\mu' \leftarrow \mathsf{Frodo.Decode}(\mathbf{M})$
5: Parse $pk \leftarrow \mathsf{seed}_\mathbf{A} \| \mathbf{b}$
6: Generate pseudorandom values $\mathsf{seed}'_\mathbf{E} \| \mathbf{k}' \| \mathbf{d}' \leftarrow G(pk \| \mu')$
7: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}'_\mathbf{E}, \overline{m}, n, T_\chi, 4)$
8: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}'_\mathbf{E}, \overline{m}, n, T_\chi, 5)$
9: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_\mathbf{A})$
10: Compute $\mathbf{B}'' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
11: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}(\mathsf{seed}'_\mathbf{E}, \overline{m}, \overline{n}, T_\chi, 6)$
12: Compute $\mathbf{B} \leftarrow \mathsf{Frodo.Unpack}(\mathbf{b}, n, \overline{n})$
13: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
14: Compute $\mathbf{C}' \leftarrow \mathbf{V} + \mathsf{Frodo.Encode}(\mu')$
15: **if** $\mathbf{B}' \| \mathbf{C} = \mathbf{B}'' \| \mathbf{C}'$ and $\mathbf{d} = \mathbf{d}'$ **then**
16: **return** shared secret $\mathbf{ss} \leftarrow F(\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{k}' \| \mathbf{d})$
17: **else**
18: **return** shared secret $\mathbf{ss} \leftarrow F(\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{s} \| \mathbf{d})$

IND-CCA secure FrodoKEM

FrodoKEM.KeyGen

FrodoKEM.Enc

FrodoKEM.Dec

FrodoKEM: “Wide enough” Gaussians

Main Design Goal of Frodo: Conservative and Practical

Challenge: LWE parameters that respect worst-case reductions

- FrodoCCS
  - Generic lattices, but narrow Gaussians
  - Need $\sigma > c\sqrt{n}$ where is c a constant—can be improved from **[Regev 05]**
  - Still not small enough for practical values

- FrodoKEM
  - Generic lattices, with “wide enough” Gaussians
  - Reduction from BDD (with Discrete Gaussian Samples)
  - Reductions where σ only depends on the smoothing parameter, number of DGS
  - Set very conservative estimates for both (s.p.=2^{-150}, N = 2^{256})

Well studied problem!

FrodoKEM: Parameter Sets

Table 1: **Parameter sets**

	n	q	σ	**support of χ**	B	$\bar{m} \times \bar{n}$	**failure prob.**	**c size (bytes)**	**Security C**	**Q**
Frodo-640	640	2^{15}	2.75	$[-11 \ldots 11]$	2	8×8	$2^{-148.8}$	9,736	143	103
Frodo-976	976	2^{16}	2.3	$[-10 \ldots 10]$	3	8×8	$2^{-199.6}$	15,768	209	150

FrodoKEM: Parameter Sets

Table 1: **Parameter sets**

	n	q	σ	**support of** χ	B	$\bar{m} \times \bar{n}$	**failure prob.**	c **size (bytes)**	**Security C**	**Q**
Frodo-640	640	2^{15}	2.75	$[-11 \ldots 11]$	2	8×8	$2^{-148.8}$	9,736	143	103
Frodo-976	976	2^{16}	2.3	$[-10 \ldots 10]$	3	8×8	$2^{-199.6}$	15,768	209	150

FrodoKEM: Parameter Sets

Table 1: **Parameter sets**

	n	q	σ	**support of** χ	B	$\bar{m} \times \bar{n}$	**failure prob.**	c **size (bytes)**	**Security** C	Q
Frodo-640	640	2^{15}	2.75	$[-11 \ldots 11]$	2	8×8	$2^{-148.8}$	9,736	143	103
Frodo-976	976	2^{16}	2.3	$[-10 \ldots 10]$	3	8×8	$2^{-199.6}$	15,768	209	150

"Paranoid estimate" based off of list-size of ***any*** sieving algorithm is even more conservative

This estimate satisfies NIST **Level1** and **Level3** targets resp.

FrodoKEM: Parameter Sets

Table 1: **Parameter sets**

	n	q	σ	**support of χ**	B	$\bar{m} \times \bar{n}$	**failure prob.**	**c size (bytes)**	**Security** C	Q
Frodo-640	640	2^{15}	2.75	$[-11 \ldots 11]$	2	8×8	$2^{-148.8}$	9,736	143	103
Frodo-976	976	2^{16}	2.3	$[-10 \ldots 10]$	3	8×8	$2^{-199.6}$	15,768	209	150

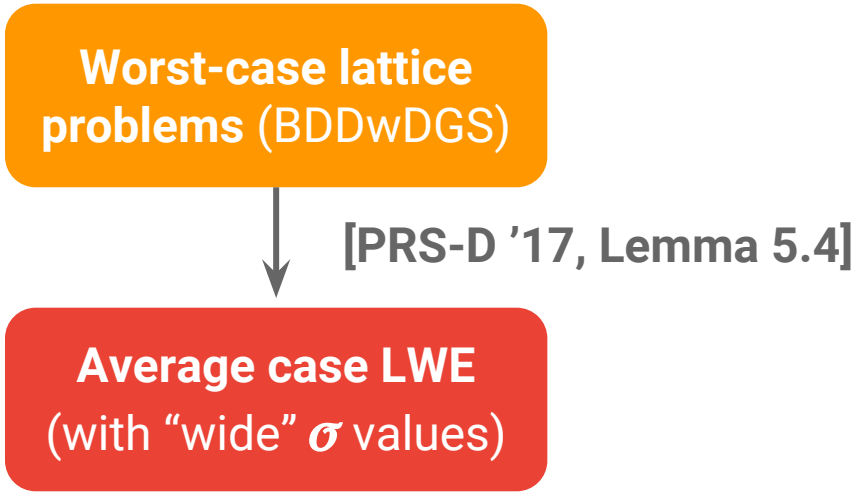
Table 2: **Error distributions**

	σ	**Probability of (in multiples of 2^{-15})** 0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10	±11	**Rényi order**	**Rényi divergence**
$\chi_{\text{Frodo-640}}$	2.75	9456	8857	7280	5249	3321	1844	898	384	144	47	13	3	500.0	0.72×10^{-4}
$\chi_{\text{Frodo-976}}$	2.3	11278	10277	7774	4882	2545	1101	396	118	29	6	1		500.0	0.14×10^{-4}

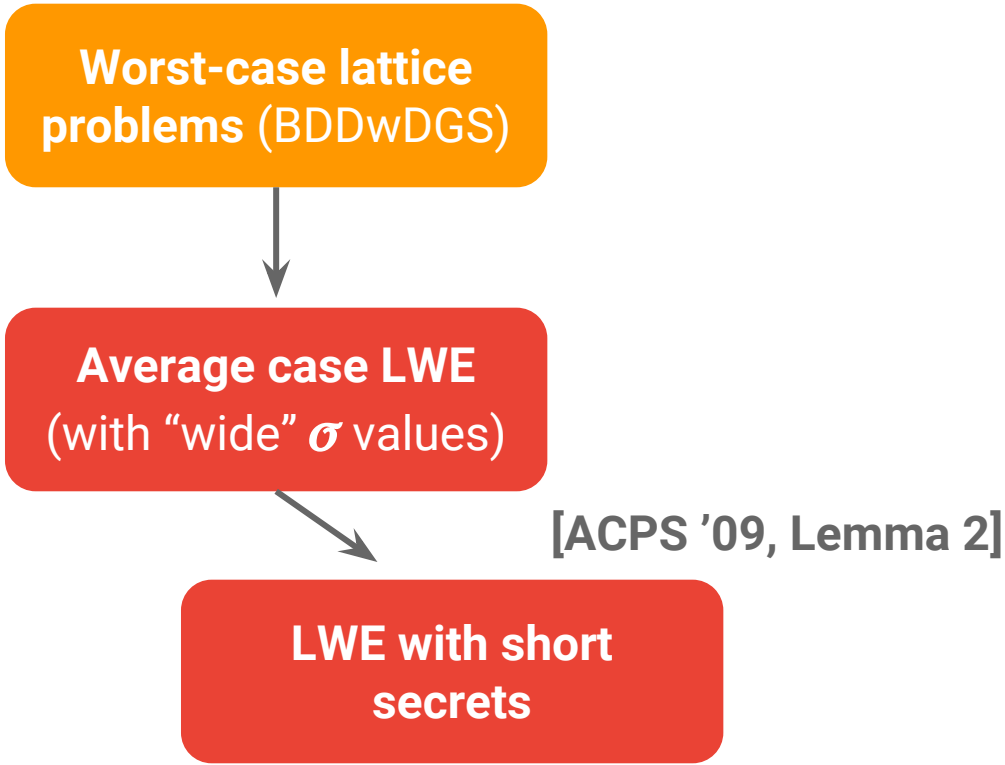
FrodoKEM: Security Reductions

Worst-case lattice problems (BDDwDGS)

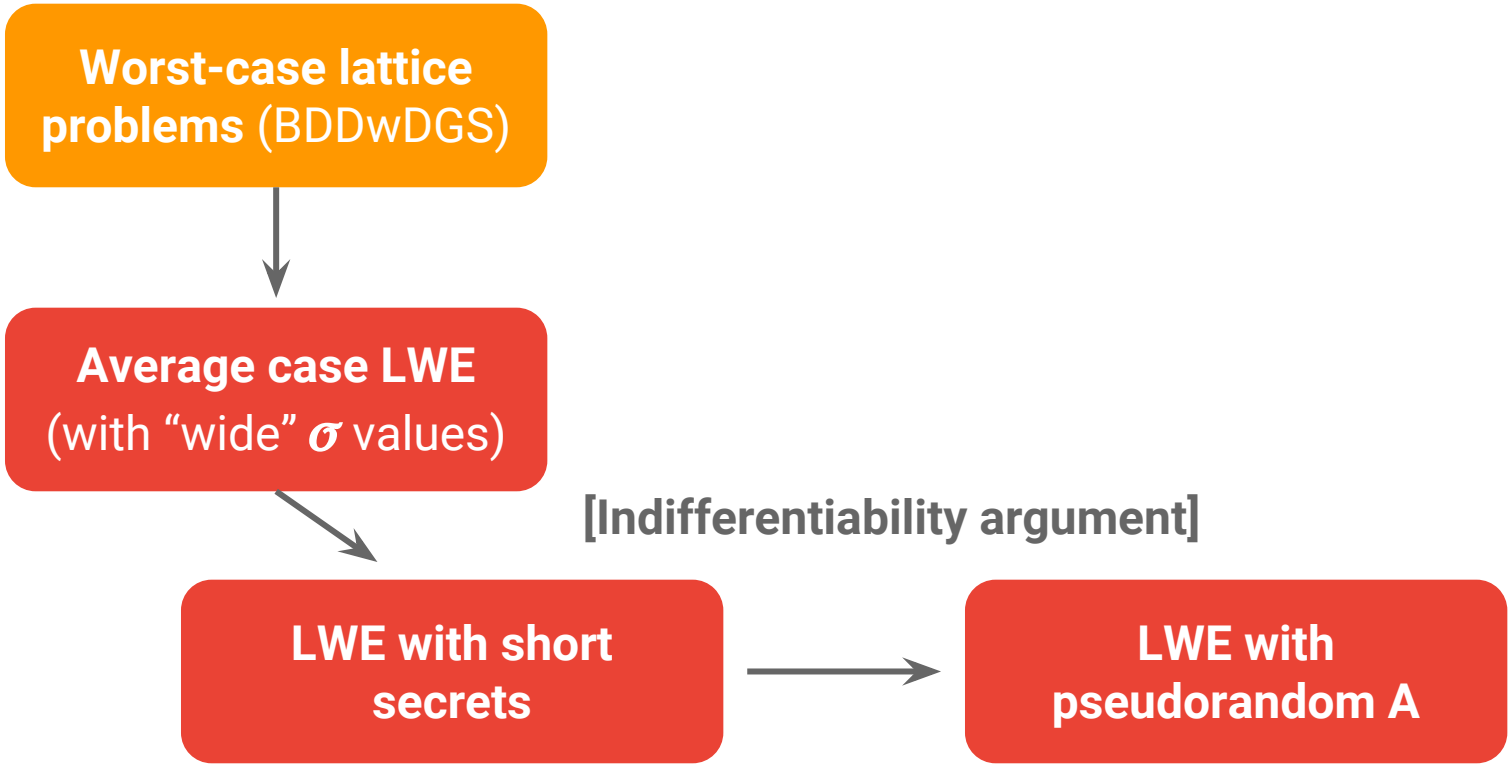
FrodoKEM: Security Reductions



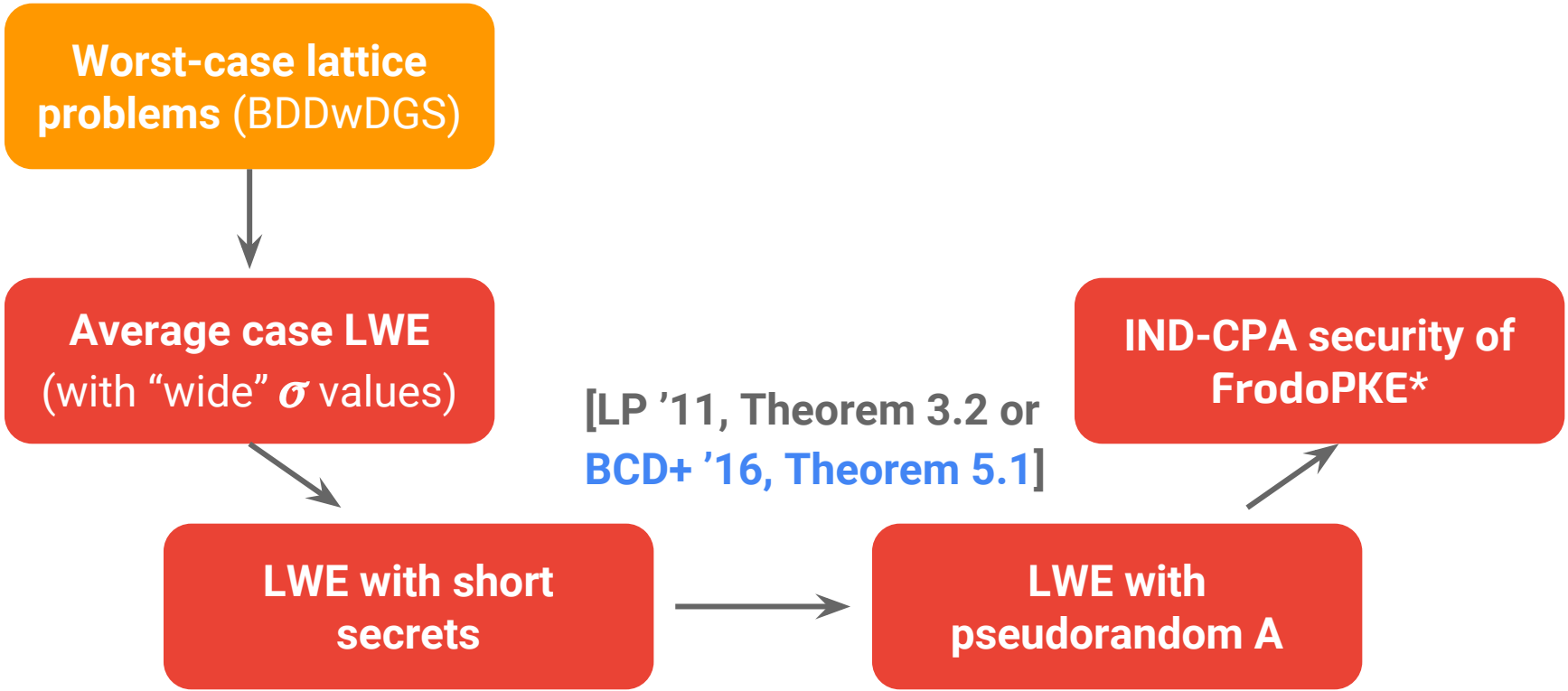
FrodoKEM: Security Reductions



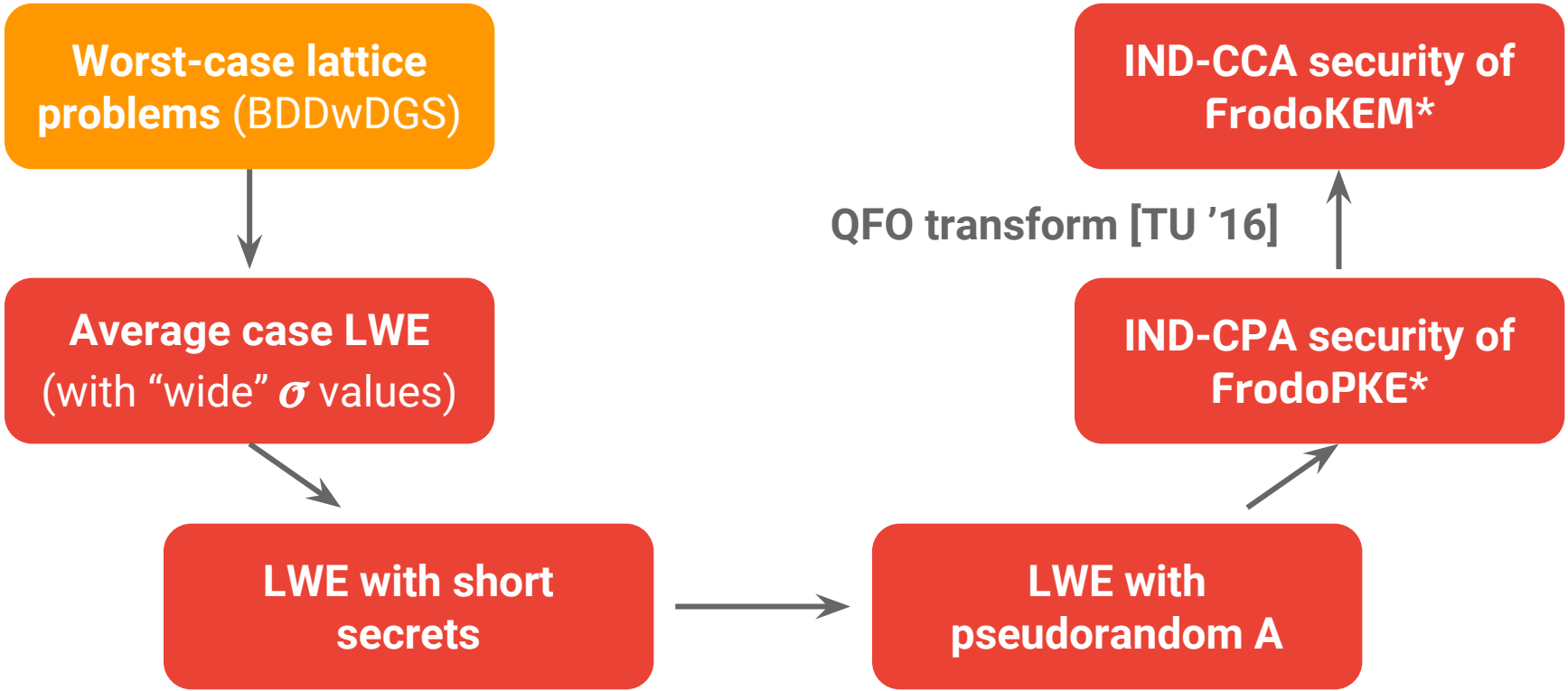
FrodoKEM: Security Reductions



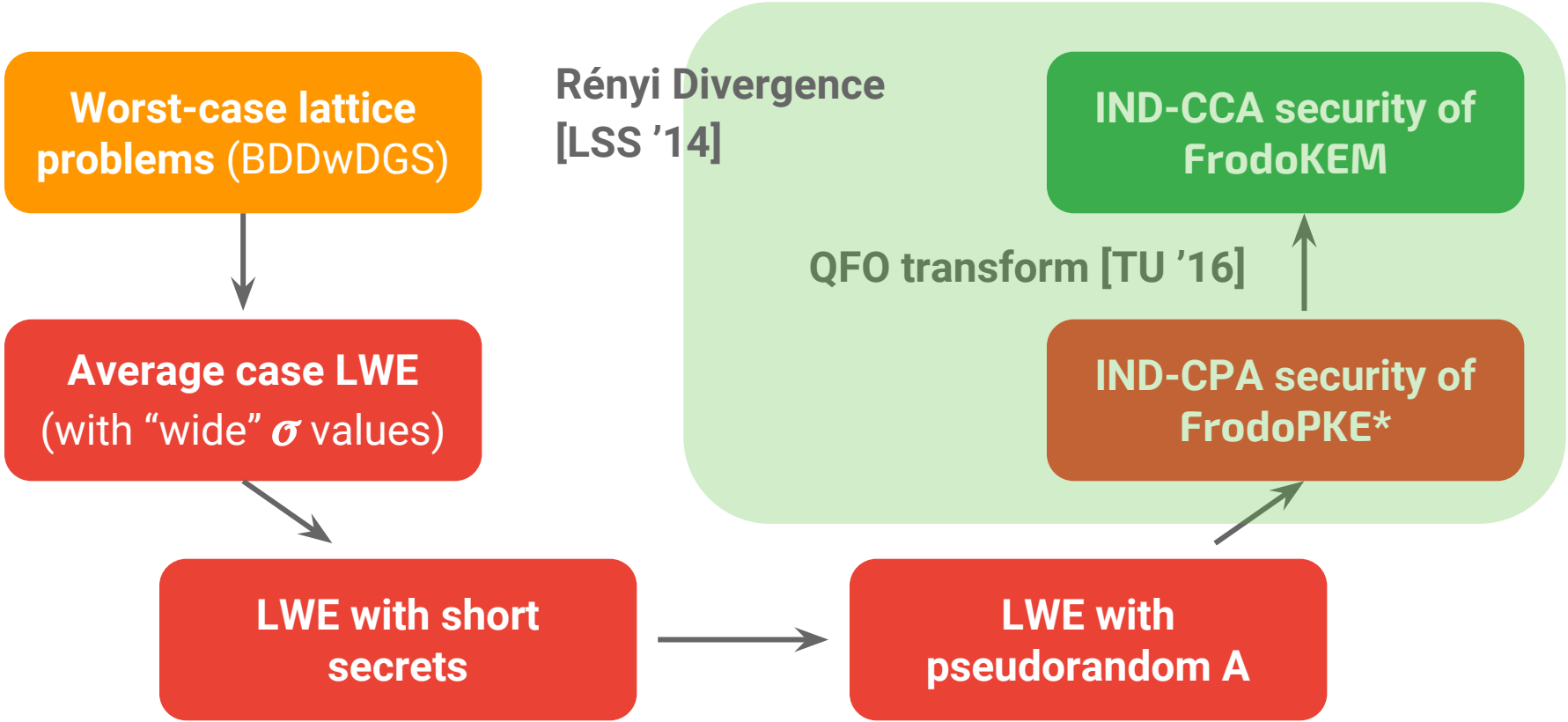
FrodoKEM: Security Reductions



FrodoKEM: Security Reductions



FrodoKEM: Security Reductions



FrodoKEM: Security Reductions

Worst-case lattice problems (BDDwDGS)

IND-CCA security of FrodoKEM

FrodoKEM: Stand-alone Benchmarks

Table 5: **Performance (in thousands of cycles) of** FrodoKEM **on a 3.4GHz Intel Core i7-6700 (Skylake) processor with matrix A generated using** AES128**.** Results are reported using OpenSSL's AES implementation and using a standalone AES implementation, all of which exploit AES-NI instructions. Cycle counts are rounded to the nearest 10^3 cycles.

Scheme	KeyGen	Encaps	Decaps	Total (Encaps + Decaps)
Optimized Implementation (AES from OpenSSL)				
FrodoKEM-640-AES	1,287	1,810	1,811	3,621 ≈ 1.1 msec
FrodoKEM-976-AES	2,715	3,572	3,588	7,160 ≈ 2.1 msec
Additional implementation using AVX2 intrinsic instructions (AES from OpenSSL)				
FrodoKEM-640-AES	1,293	1,828	1,829	3,657
FrodoKEM-976-AES	2,663	3,565	3,580	7,145
Additional implementation using AVX2 intrinsic instructions (standalone AES)				
FrodoKEM-640-AES	1,288	1,834	1,837	3,671
FrodoKEM-976-AES	2,677	3,577	3,580	7,157

FrodoKEM: Key Sizes

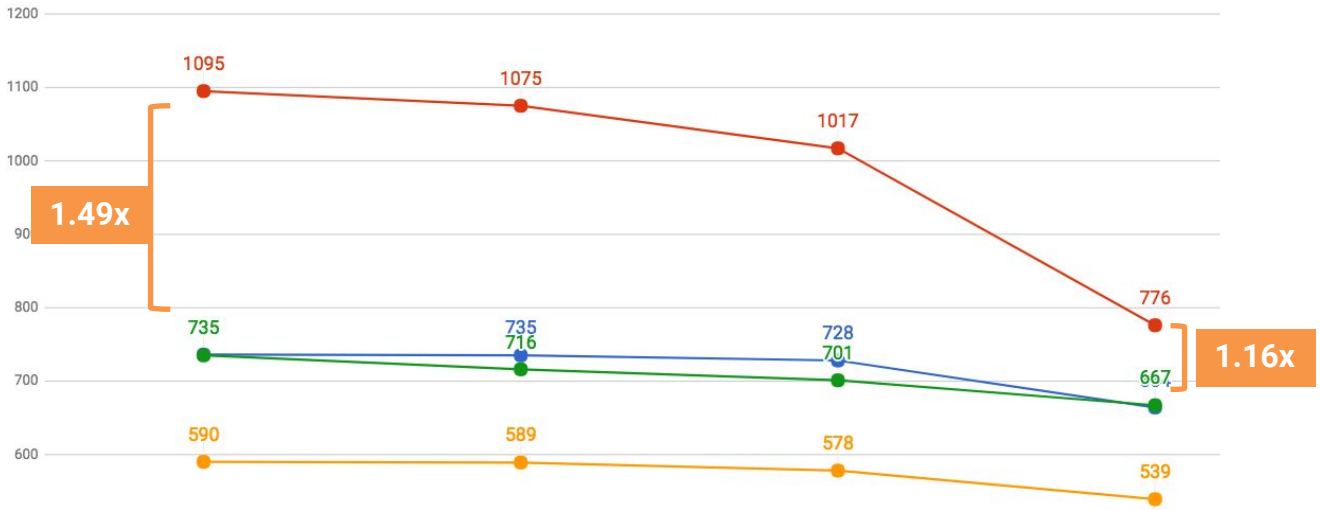
Table 4: **Size (in bytes) of inputs and outputs of** FrodoKEM. Secret key size is the sum of the sizes of the actual secret value and of the public key (the NIST API does not include the public key as explicit input to decapsulation).

Scheme	secret key *sk*	public key *pk*	ciphertext *c*	shared secret ss
FrodoKEM-640	19,872 (10,256 + 9,616)	9,616	9,736	16
FrodoKEM-976	31,272 (15,640 + 15,632)	15,632	15,768	24

FrodoKEM: Summary

Practical

Encryption ≈ 0.5 ms
Decryption ≈ 0.5 ms

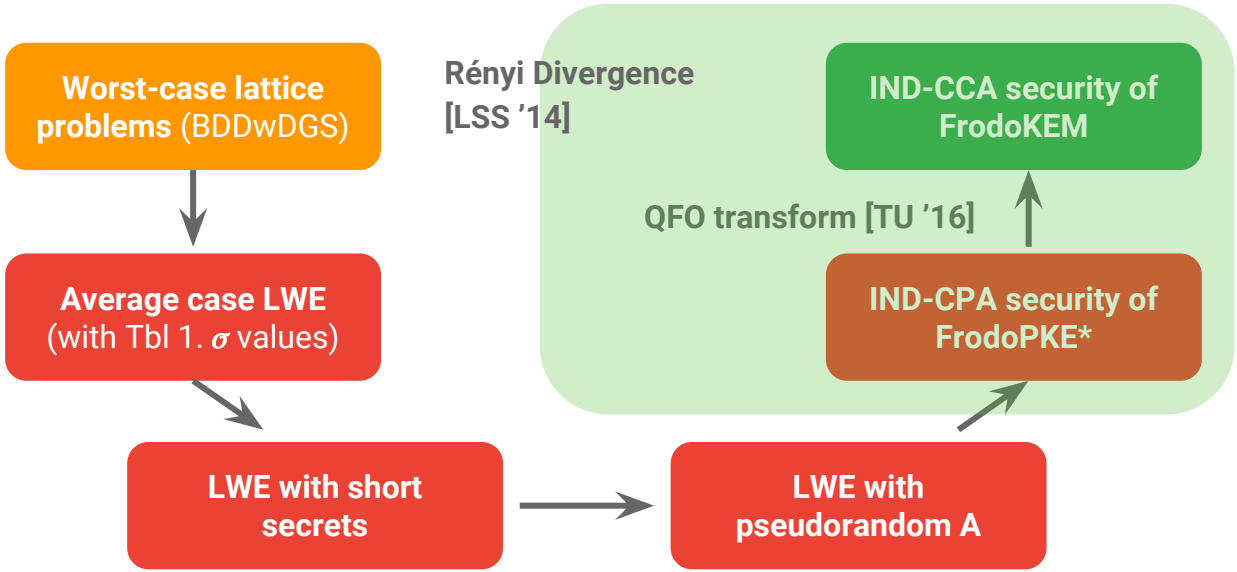


FrodoKEM: Summary

Practical Quantum-Secure

FrodoKEM: Summary

Practical Quantum-Secure



FrodoKEM: Summary

Practical Quantum-Secure Key Encapsulation

Algorithm 12 FrodoKEM.KeyGen.

Input: None.

Output: Key pair (pk, sk') with $pk \in \{0,1\}^{\mathsf{len_A}+D\cdot n\cdot\overline{n}}$, $sk' \in \{0,1\}^{\mathsf{len_s}+\mathsf{len_A}+D\cdot n\cdot\overline{n}} \times \mathbb{Z}_q^{n\times\overline{n}}$.

1: Choose
2: Genera
3: Genera
4: Sample
5: Sample
6: Compu
7: Compu
8: **return**

Algorithm 13 FrodoKEM.Encaps.

Input: Public key $pk = \mathsf{seed_A}\|\mathbf{b} \in \{0,1\}^{\mathsf{len_A}+D\cdot n\cdot\overline{n}}$

Output:

1: Choos
2: Gener
3: Sampl
4: Sampl
5: Gener
6: Comp
7: Comp
8: Sampl
9: Comp
10: Comp
11: Comp
12: Comp
13: Comp
14: retur

Algorithm 14 FrodoKEM.Decaps.

Input: Ciphertext $\mathbf{c}_1\|\mathbf{c}_2\|\mathbf{d} \in \{0,1\}^{(\overline{m}\cdot n+\overline{m}\cdot\overline{n})D+\mathsf{len_d}}$, secret key $sk' = (\mathbf{s}\|\mathsf{seed_A}\|\mathbf{b},\mathbf{S}) \in \{0,1\}^{\mathsf{len_s}} \times \mathbb{Z}_q^{n\times\overline{n}}$.

Output: Shared secret $\mathbf{ss} \in \{0,1\}^{\mathsf{len_{ss}}}$.

1: $\mathbf{B}' \leftarrow$ Frodo.Unpack($\mathbf{c}_1$)
2: $\mathbf{C} \leftarrow$ Frodo.Unpack($\mathbf{c}_2$)
3: Compute $\mathbf{M} \leftarrow \mathbf{C} - \mathbf{B}'\mathbf{S}$
4: Compute $\mu' \leftarrow$ Frodo.Decode($\mathbf{M}$)
5: Parse $pk \leftarrow \mathsf{seed_A}\|\mathbf{b}$
6: Generate pseudorandom values $\mathsf{seed}'_{\mathbf{E}}\|\mathbf{k}'\|\mathbf{d}' \leftarrow G(pk\|\mu')$
7: Sample error matrix $\mathbf{S}' \leftarrow$ Frodo.SampleMatrix($\mathsf{seed}'_{\mathbf{E}}, \overline{m}, n, T_\chi, 4$)
8: Sample error matrix $\mathbf{E}' \leftarrow$ Frodo.SampleMatrix($\mathsf{seed}'_{\mathbf{E}}, \overline{m}, n, T_\chi, 5$)
9: Generate $\mathbf{A} \leftarrow$ Frodo.Gen($\mathsf{seed_A}$)
10: Compute $\mathbf{B}'' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
11: Sample error matrix $\mathbf{E}'' \leftarrow$ Frodo.SampleMatrix($\mathsf{seed}'_{\mathbf{E}}, \overline{m}, \overline{n}, T_\chi, 6$)
12: Compute $\mathbf{B} \leftarrow$ Frodo.Unpack($\mathbf{b}, n, \overline{n}$)
13: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
14: Compute $\mathbf{C}' \leftarrow \mathbf{V} +$ Frodo.Encode(μ')
15: **if** $\mathbf{B}'\|\mathbf{C} = \mathbf{B}''\|\mathbf{C}'$ and $\mathbf{d} = \mathbf{d}'$ **then**
16: **return** shared secret $\mathbf{ss} \leftarrow F(\mathbf{c}_1\|\mathbf{c}_2\|\mathbf{k}'\|\mathbf{d})$
17: **else**
18: **return** shared secret $\mathbf{ss} \leftarrow F(\mathbf{c}_1\|\mathbf{c}_2\|\mathbf{s}\|\mathbf{d})$

FrodoKEM.KeyGen

FrodoKEM.Enc

FrodoKEM.Dec

FrodoKEM: Summary

Practical Quantum-Secure Key Encapsulation from (Generic) Lattices

FrodoKEM: Summary

Practical Quantum-Secure Key Encapsulation from (Generic) Lattices

Thank You!

Any Questions?

pseudorandom@google.com, team@frodokem.org