Post-quantum key exchange for the TLS protocol from the ring learning with errors problem

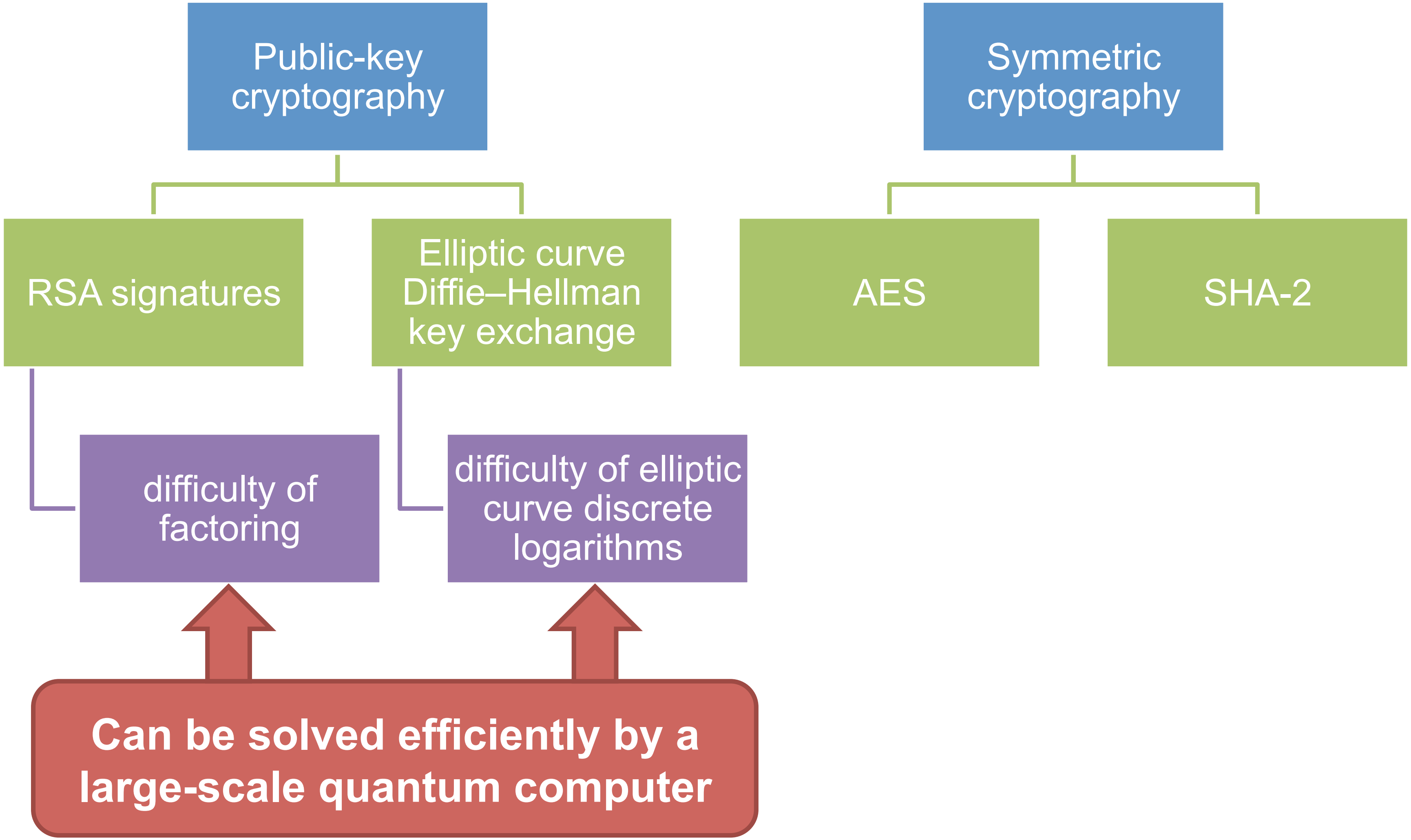
Joppe W. Bos *(NXP Semiconductors)*,
Craig Costello & **Michael Naehrig** *(Microsoft Research)*,
Douglas Stebila *(Queensland University of Technology)*

IEEE Symposium on Security & Privacy (S&P) 2015 • May 19, 2015

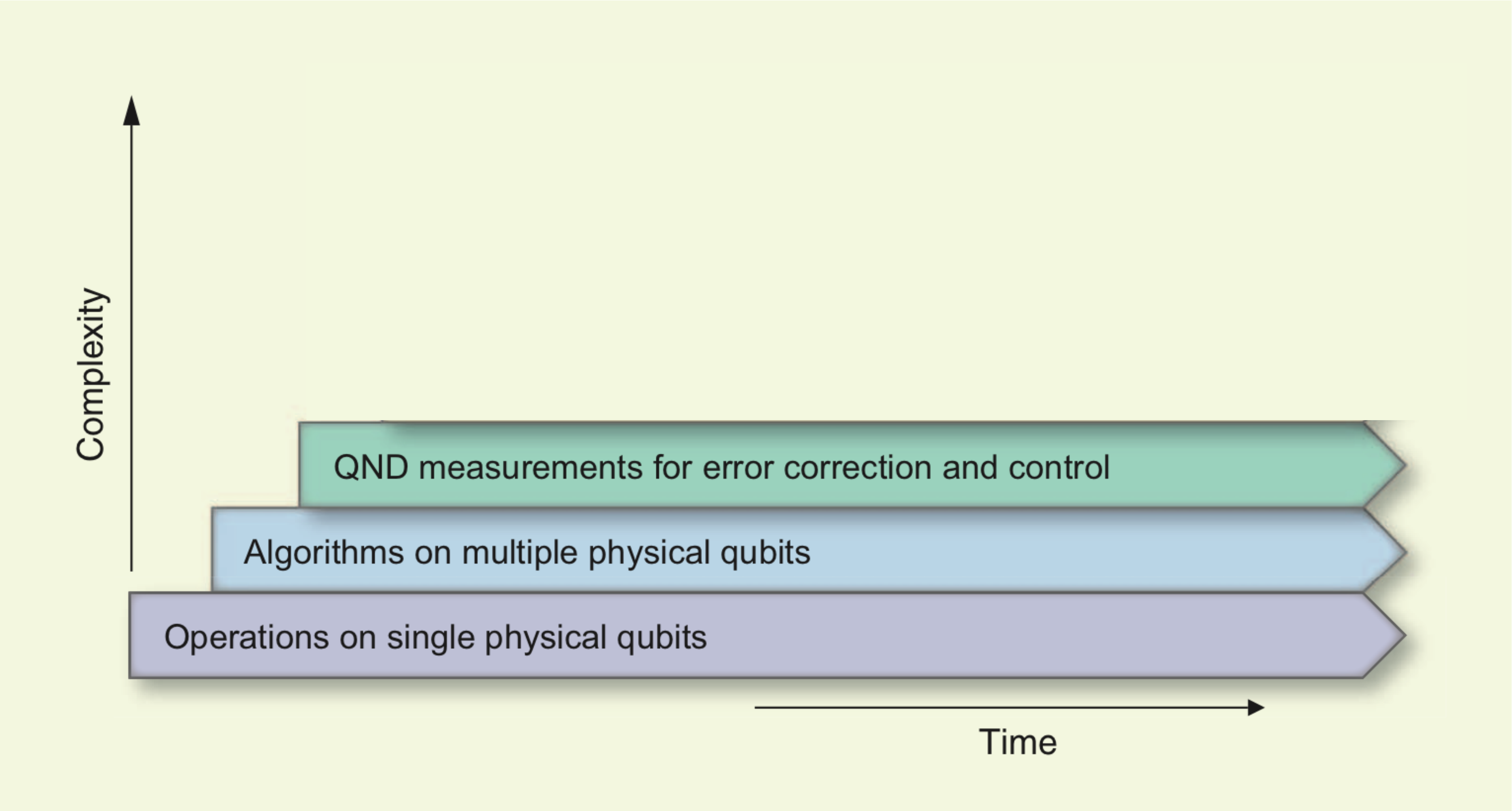
① Motivation

Contemporary cryptography

TLS-ECDHE-RSA-AES128-GCM-SHA256

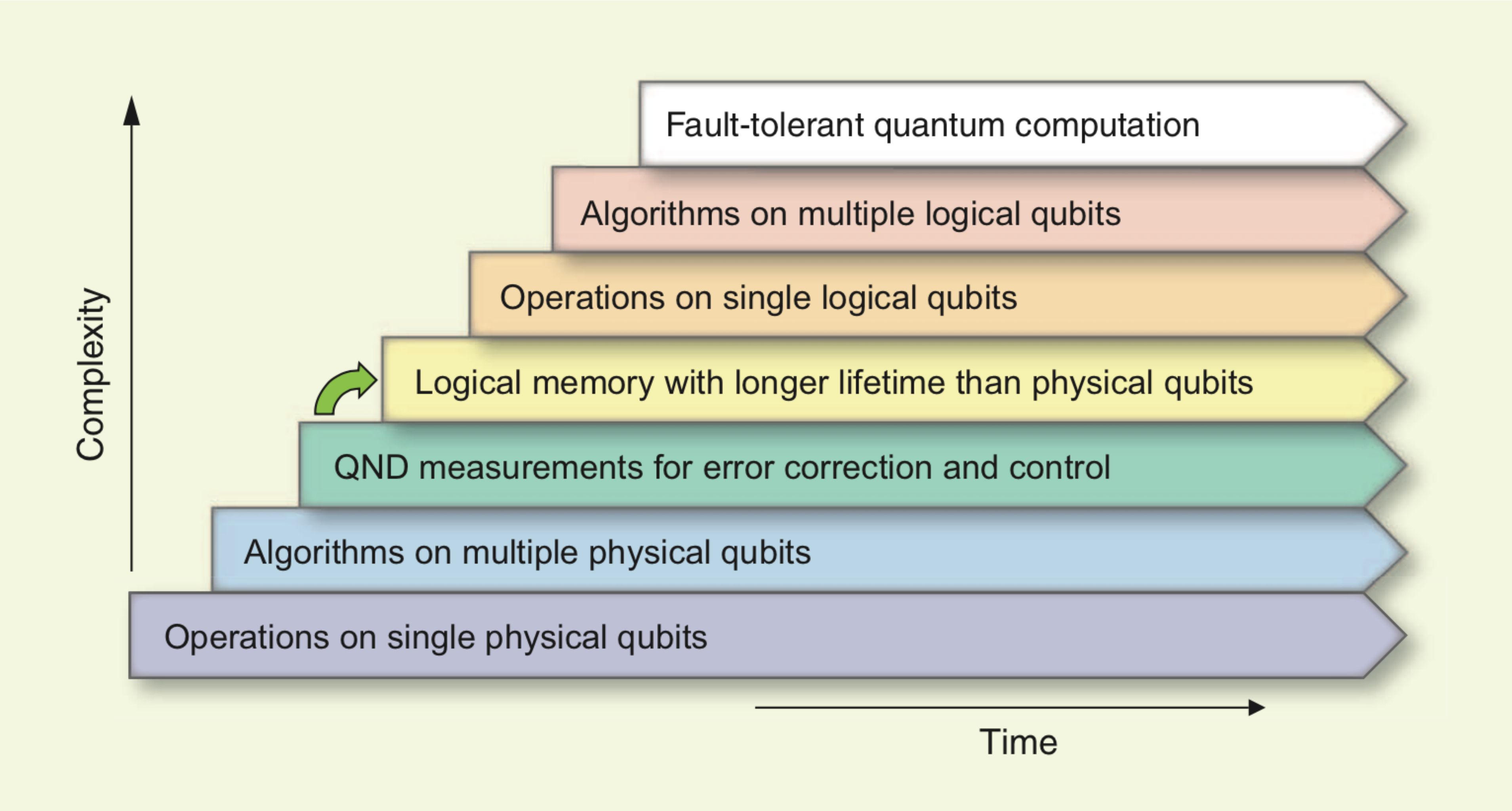


Building quantum computers



Devoret, Schoelkopf. *Science* 339:1169–1174, March 2013.

Building quantum computers



Devoret, Schoelkopf. *Science* 339:1169–1174, March 2013.

Post-quantum / quantum-safe crypto

No known exponential quantum speedup:

Code-based

- McEliece

Hash-based

- Merkle signatures
- Sphincs

Multivariate

- multivariate quadratic

Lattice-based

- NTRU
- learning with errors
- ring-LWE

Lots of questions

- Better classical or quantum attacks on post-quantum schemes?
- What are the right parameter sizes?
- Are the key sizes sufficiently small?
- Can we do the operations sufficiently fast?
- How do we integrate them into the existing infrastructure?

Lots of questions

This talk: ring learning with errors

- Are the key sizes sufficiently small?
- Can we do the operations sufficiently fast?
- How do we integrate them into the existing infrastructure?

This talk: ring-LWE key agreement in TLS

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

- Signatures still done with traditional primitives (RSA/ECDSA)
  - we only need authentication to be secure *now*
  - benefit: use existing RSA-based PKI
- Key agreement done with ring-LWE

② Learning with errors

Solving systems of linear equations

random $\mathbb{Z}_{13}^{7\times 4}$ **secret** $\mathbb{Z}_{13}^{4\times 1}$ $\mathbb{Z}_{13}^{7\times 1}$

$$\begin{pmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{pmatrix} \times \begin{pmatrix} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} 4 \\ 8 \\ 1 \\ 10 \\ 4 \\ 12 \\ 9 \end{pmatrix}$$

Linear system problem: given **blue**, find **red**

Solving systems of linear equations

random $\mathbb{Z}_{13}^{7\times 4}$ **secret** $\mathbb{Z}_{13}^{4\times 1}$ $\mathbb{Z}_{13}^{7\times 1}$

$$\begin{pmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{pmatrix} \times \begin{pmatrix} 6 \\ 9 \\ 11 \\ 11 \end{pmatrix} = \begin{pmatrix} 4 \\ 8 \\ 1 \\ 10 \\ 4 \\ 12 \\ 9 \end{pmatrix}$$

Easily solved using Gaussian elimination (Linear Algebra 101)

Linear system problem: given **blue**, find **red**

Learning with errors problem

$$\underset{\substack{\textbf{random}\\ \mathbb{Z}_{13}^{7\times 4}}}{\begin{pmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{pmatrix}} \times \underset{\substack{\textbf{secret}\\ \mathbb{Z}_{13}^{4\times 1}}}{\begin{pmatrix} 6 \\ 9 \\ 11 \\ 11 \end{pmatrix}} + \underset{\substack{\textbf{small noise}\\ \mathbb{Z}_{13}^{7\times 1}}}{\begin{pmatrix} 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \\ -1 \end{pmatrix}} = \underset{\mathbb{Z}_{13}^{7\times 1}}{\begin{pmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{pmatrix}}$$

Learning with errors problem

random $\mathbb{Z}_{13}^{7\times 4}$ **secret** $\mathbb{Z}_{13}^{4\times 1}$ **small noise** $\mathbb{Z}_{13}^{7\times 1}$ $\mathbb{Z}_{13}^{7\times 1}$

$$\begin{pmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{pmatrix} \times \begin{pmatrix} \, \\ \, \\ \, \\ \, \end{pmatrix} + \begin{pmatrix} \, \\ \, \\ \, \\ \, \\ \, \\ \, \\ \, \end{pmatrix} = \begin{pmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{pmatrix}$$

LWE problem: given **blue**, find **red**

Toy example versus real-world example

$\mathbb{Z}_{13}^{7\times 4}$

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

$\mathbb{Z}_{4093}^{640\times 256}$

256 (columns) × 640 (rows)

2738	3842	3345	2979	…
2896	595	3607		
377	1575			
2760				
…				

640 × 256 × 12 bits = **245 KiB**

Ring learning with errors problem

random
$\mathbb{Z}_{13}^{7\times 4}$

4	1	11	10
10	4	1	11
11	10	4	1
1	11	10	4
4	1	11	10
10	4	1	11
11	10	4	1

Each row is the cyclic shift of the row above

Ring learning with errors problem

random
$\mathbb{Z}_{13}^{7\times 4}$

4	1	11	10
3	4	1	11
2	3	4	1
12	2	3	4
9	12	2	3
10	9	12	2
11	10	9	12

Each row is the cyclic shift of the row above

…

with a special wrapping rule: *x* wraps to –*x* mod 13.

Ring learning with errors problem

random

$\mathbb{Z}_{13}^{7\times 4}$

4	**1**	**11**	**10**

Each row is the cyclic
shift of the row above
…
with a special wrapping rule:
x wraps to –*x* mod 13.

So I only need to tell you the first row.

Ring learning with errors problem

$$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$$

	$4 + 1x + 11x^2 + 10x^3$	**random**
×	$6 + 9x + 11x^2 + 11x^3$	**secret**
+	$0 - 1x + 1x^2 + 1x^3$	**small noise**
=	$10 + 5x + 10x^2 + 7x^3$	

Ring learning with errors problem

$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$

	$4 + 1x + 11x^2 + 10x^3$	**random**
×		**secret**
+		**small noise**
=	**$10 + 5x + 10x^2 + 7x^3$**	

Ring-LWE problem: given **blue**, find **red**

Decision ring learning with errors problem with small secrets

$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$

	$4 + 1x + 11x^2 + 10x^3$	random
×	$1 + 0x - 1x^2 + 2x^3$	small secret
+	$0 - 1x + 1x^2 + 1x^3$	small noise
=	$10 + 5x + 10x^2 + 7x^3$	looks random

Decision ring-LWE problem: given **blue**, distinguish **green** from random

Hardness of DRLWE

Theory:

- Poly-time (quantum) reduction from approximate shortest-independent vector problem (SIVP) on ideal lattices in *R* to DRLWE. [LPR10]

Practice:

- Assume the best way to solve DRLWE is to solve LWE.
- Solving LWE generally involves a lattice reduction problem.
- Albrecht et al. (eprint 2015/046) have hardness estimates.

For 160-bit classical security (≥ 80-bit quantum security), need larger polynomials with larger coefficients.

$$\mathbb{Z}_{2^{32}-1}[x]/\langle x^{1024}+1\rangle$$

1024 × 32 bits = **4 KiB**

③ Key agreement

Basic ring-LWE-DH key agreement (unauthenticated)

- Reformulation of Peikert's R-LWE KEM (*PQCrypto 2014*)

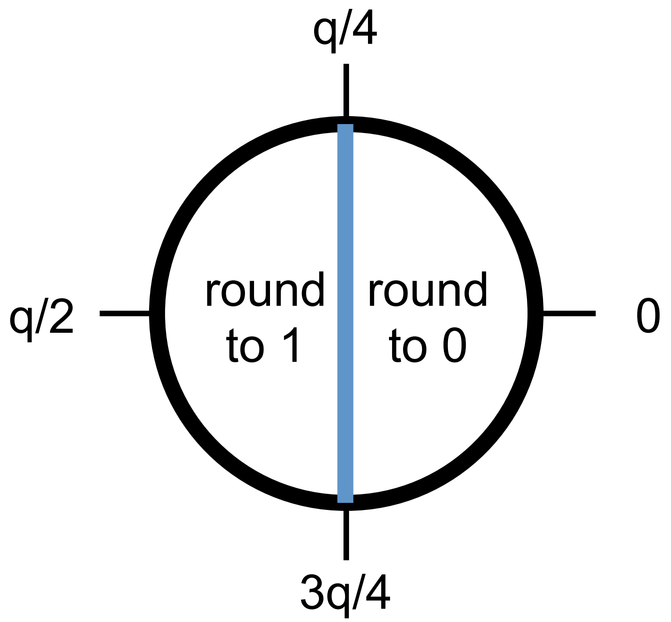
public: "big" a in $R_q = \mathbf{Z}_q[x]/(x^n+1)$

Alice		**Bob**
secret: random "small" s, e in R_q		secret: random "small" s', e' in R_q
	$b = a \bullet s + e$ →	
	← $b' = a \bullet s' + e'$	
shared secret: $s \bullet b' = s \bullet (a \bullet s' \bullet e') \approx s \bullet a \bullet s'$		shared secret: $b \bullet s' \approx s \bullet a \bullet s'$

These are only approximately equal => need rounding

Basic rounding

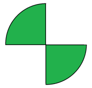
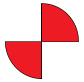
- Each coefficient of the polynomial is an integer modulo *q*
- Round either to 0 or q/2
- Treat q/2 as 1

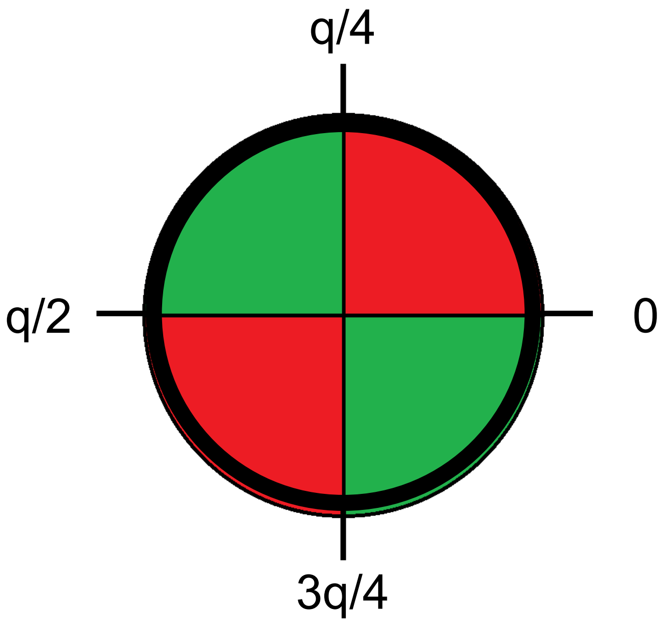


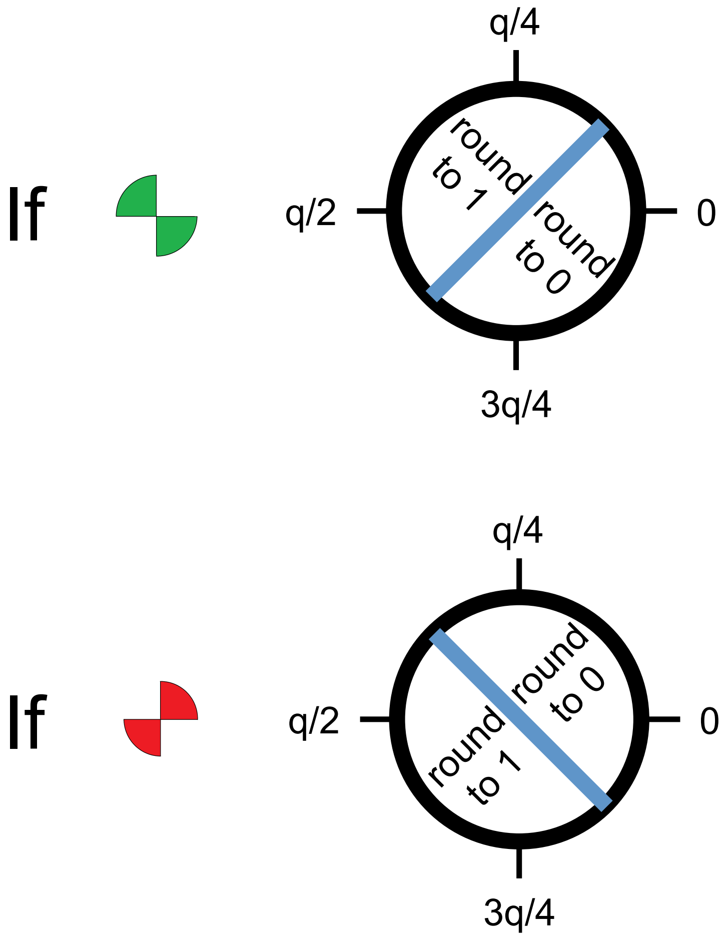
This works most of the time: prob. failure $1/2^{10}$.

Not good enough: we need exact key agreement.

Better rounding (Peikert)

- Bob says which of two regions the value is in:  or 

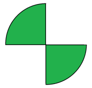
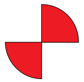




Better rounding (Peikert)

- If |u–v| ≤ q/8, then this always works.

- For our parameters, probability |u-v| > q/8 is less than $2^{-128000}$.

- Security not affected: revealing  or  leaks no information

Exact ring-LWE-DH key agreement (unauthenticated)

- Reformulation of Peikert's R-LWE KEM (*PQCrypto 2014*)

Alice

secret:
randon

s', e' *in* R_q

Secure if decision ring learning with errors problem is hard.

Decision ring-LWE is hard if a related lattice shortest vector problem is hard.

shared secret:
round($s \bullet b'$)

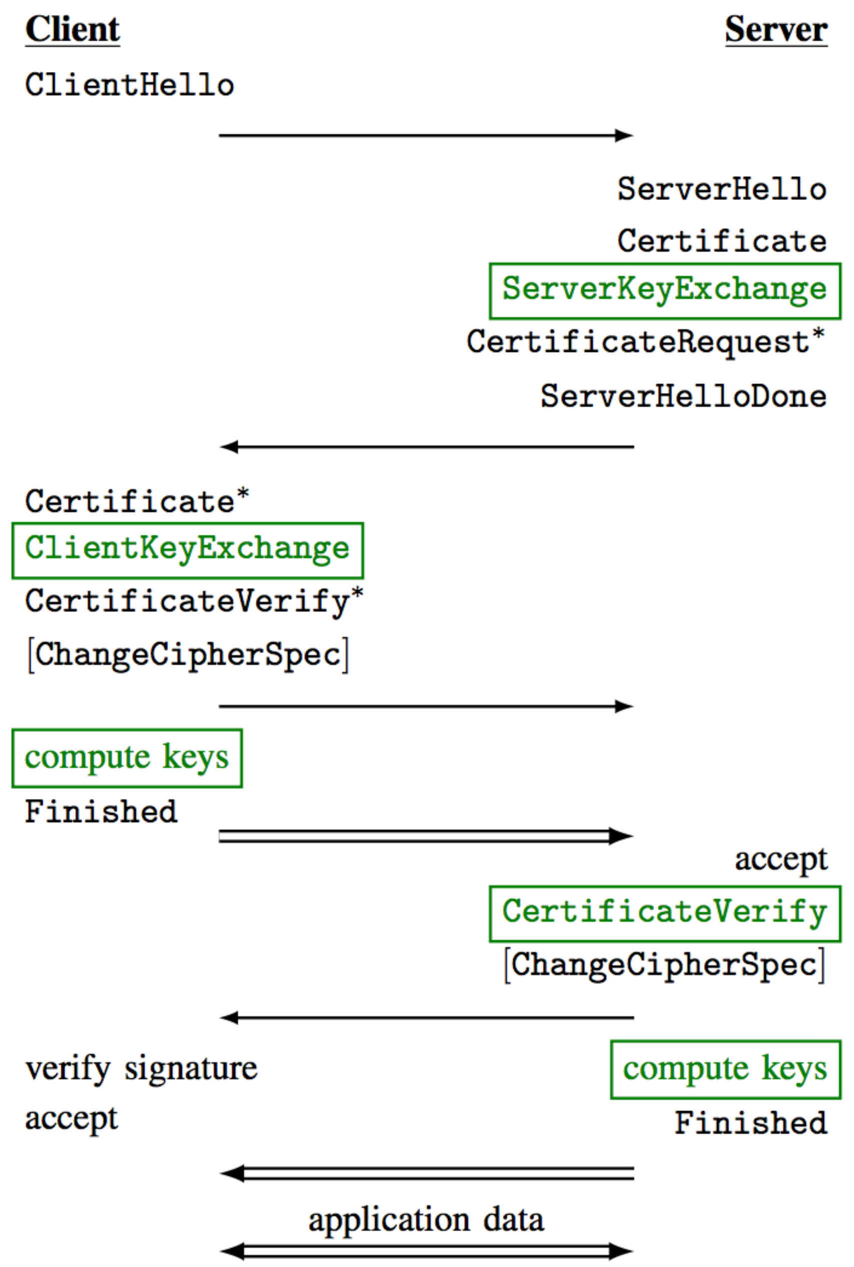
shared secret:
round($b \bullet s'$)

④ Implementation in TLS

Integration into TLS 1.2

New ciphersuite:
`TLS-RLWE-SIG-AES128-GCM-SHA256`

- RSA / ECDSA signatures for authentication
- Ring-LWE-DH for key exchange
- AES for authenticated encryption



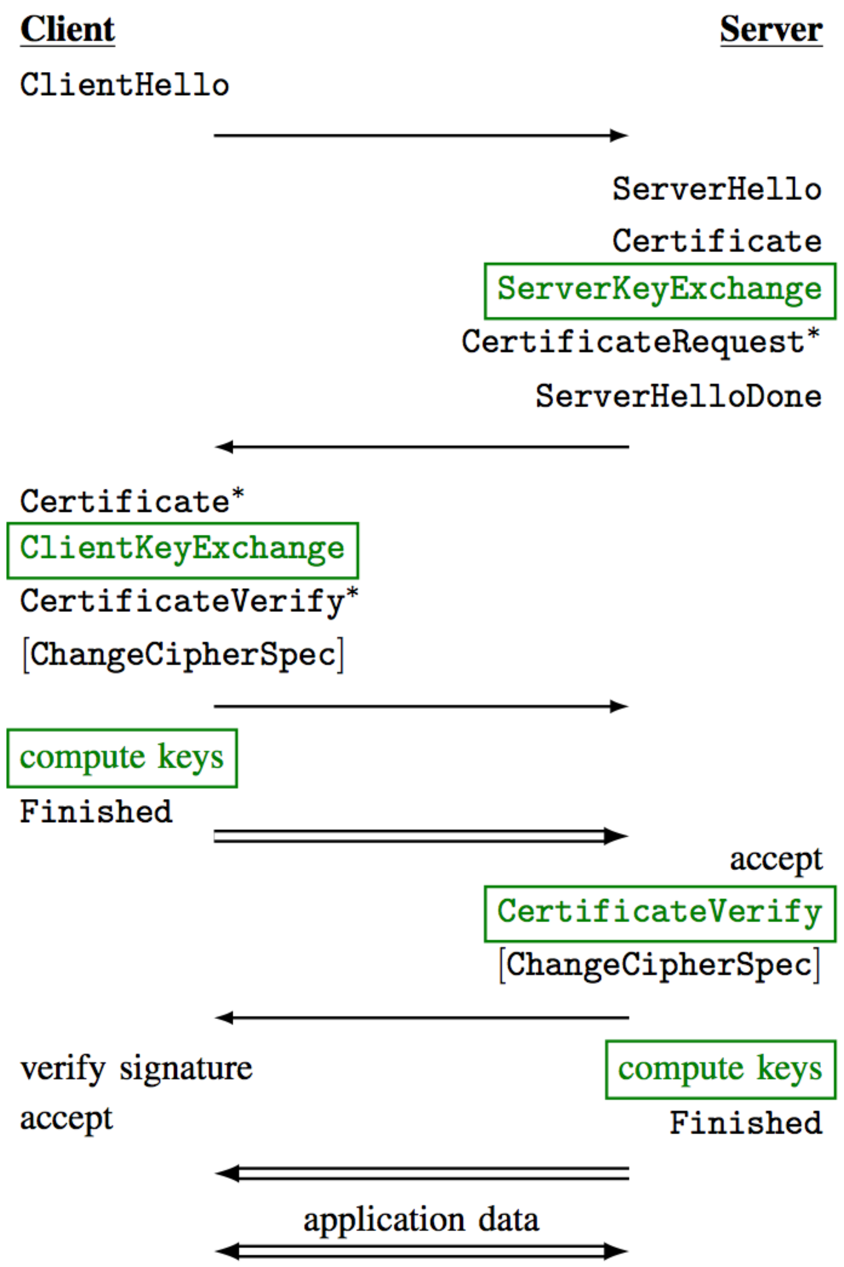
Security within TLS 1.2

Model:

- authenticated and confidential channel establishment (ACCE) (Jager et al., *Crypto 2012*)

Theorem:

- signed ring-LWE ciphersuite is ACCE-secure if underlying primitives (signatures, ring-LWE, authenticated encryption) are secure
  - Interesting technical detail for ACCE provable security people: need to move server's signature to end of TLS handshake because oracle-DH assumptions don't hold for ring-LWE



Implementation

Added ciphersuites in OpenSSL libssl

Wrapped RLWE key exchange into OpenSSL libcrypto

Basic RLWE implemented in standalone C

constant-time

non-constant-time

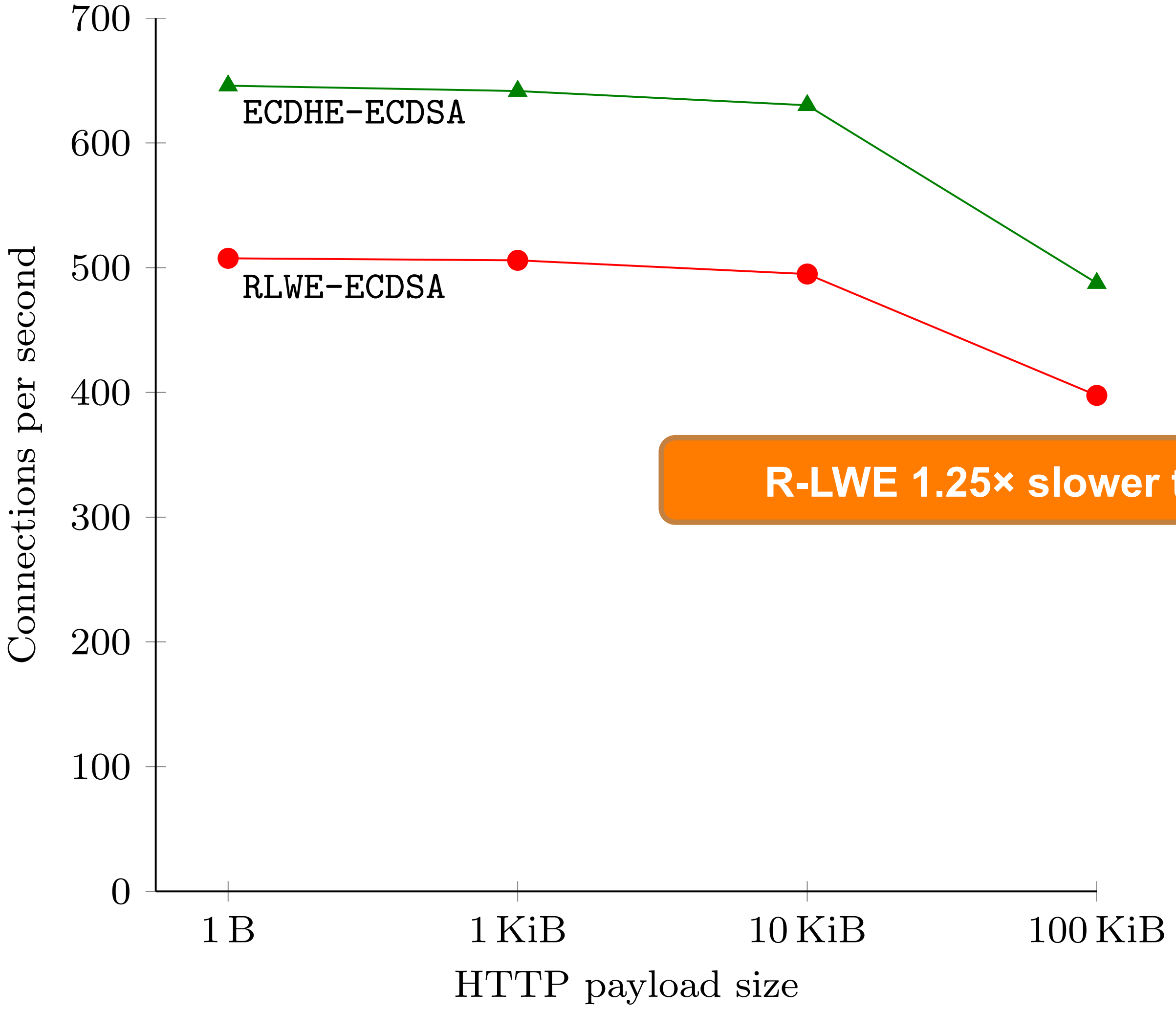
Performance – crypto operations

Operation	Client	Server
R-LWE key generation	0.9ms	0.9ms
R-LWE Alice	0.5ms	
R-LWE Bob		0.1ms
R-LWE total runtime	**1.4ms**	**1.0ms**
ECDH nistp256 (OpenSSL)	**0.8ms**	**0.8ms**

R-LWE 1.75× slower than ECDH

constant-time implementation
Intel Core i5 (4570R), 4 cores @ 2.7 GHz
llvm 5.1 (clang 503.0.30) –O3
OpenSSL 1.0.1f

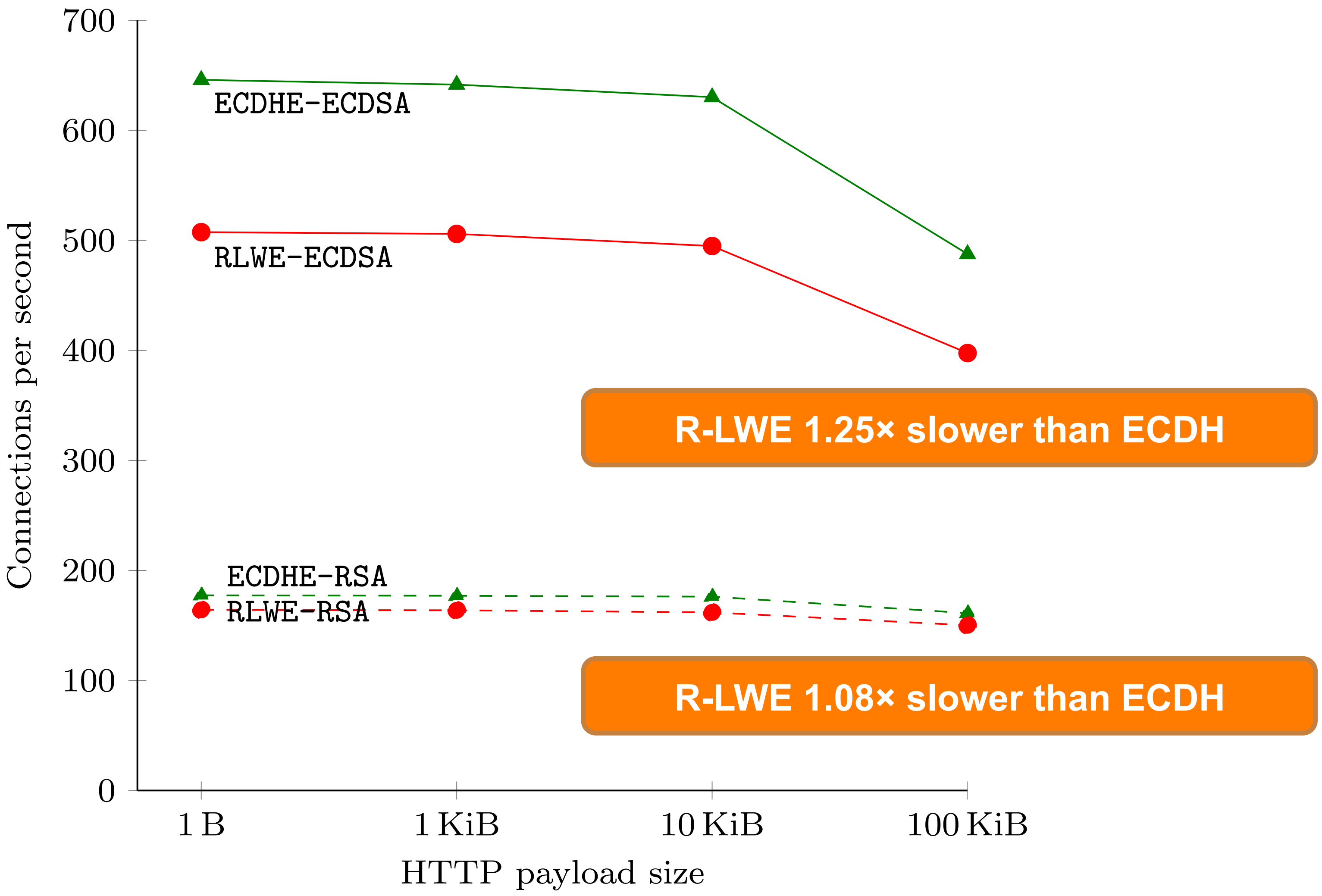
Performance – in TLS



R-LWE 1.25× slower than ECDH

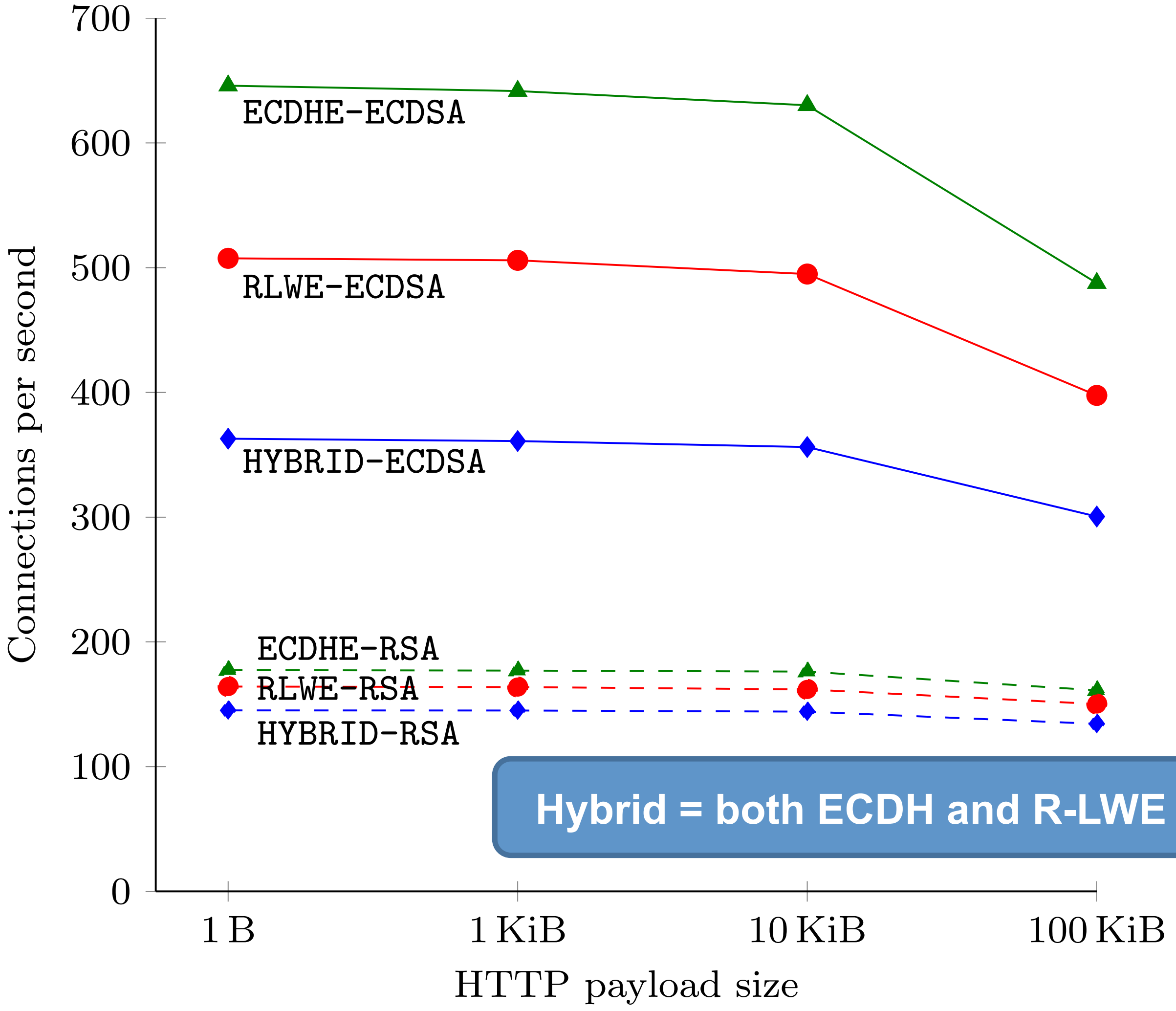
Ring-LWE adds about 8 KiB to handshake size

Performance – in TLS



Ring-LWE adds about 8 KiB to handshake size

Performance – in TLS



Hybrid = both ECDH and R-LWE key exchange

Ring-LWE adds about 8 KiB to handshake size

⑤ Summary

Summary

Ring-LWE ciphersuite with traditional signatures:

- Key sizes: not too bad (8 KiB overhead)
- Performance: small overhead (1.1–1.25×) within TLS.
- Integration into TLS: requires reordering messages, but otherwise okay.

Caveat: lattice-based assumptions less studied, algorithms solving ring-LWE may improve, security parameter estimation may evolve.

Future work

better attacks / parameter estimation

- taking into account reduction tightness
- estimate based on best quantum algorithm for solving RLWE

ring-LWE performance improvements

- assembly
- alternative FFT
- better sampling, …

other post-quantum key exchange algorithms

post-quantum authentication

Links

Full version

- http://eprint.iacr.org/2014/599

Magma code:

- http://research.microsoft.com/en-US/downloads/6bd592d7-cf8a-4445-b736-1fc39885dc6e/default.aspx

Standalone C implementation

- https://github.com/dstebila/rlwekex

Integration into OpenSSL

- https://github.com/dstebila/openssl-rlwekex