Post-quantum key exchange for the TLS protocol from R-LWE

Craig Costello
MSR-T Security and Cryptography Group

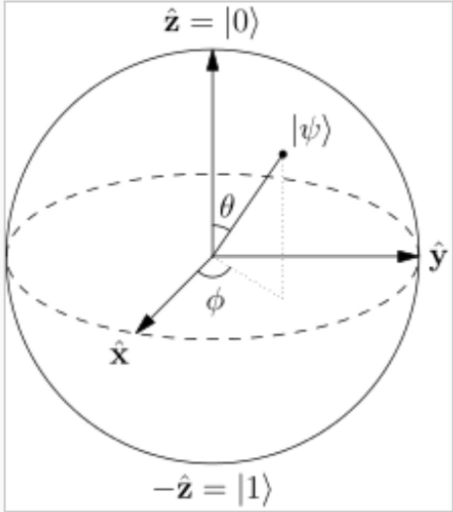
joint work with
Joppe Bos (NXP), Michael Naehrig (MSR), Douglas Stebila (QUT)

Microsoft Research

Quantum computer

From Wikipedia, the free encyclopedia

A **quantum computer** is a computation system that makes direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data.[1] Quantum computers are different from digital computers based on transistors. Whereas digital computers require data to be encoded into binary digits (bits), each of which is always in one of two definite states (0 or 1), quantum computation uses qubits (quantum bits), which can be in superpositions of states. A theoretical model is the quantum Turing machine, also known as the universal quantum computer. Quantum computers share theoretical similarities with non-deterministic and probabilistic computers; one example is the ability to be in more than one state simultaneously. The field of quantum computing was first introduced by Yuri Manin in 1980[2] and Richard Feynman in 1982.[3][4] A quantum computer with spins as quantum bits was also formulated for use as a quantum space–time in 1968.[5]



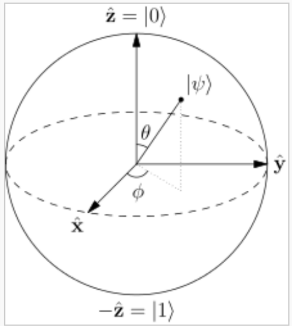
The Bloch sphere is a representation of a qubit, the fundamental building block of quantum computers.

As of 2014 quantum computing is still in its infancy but experiments have been carried out in which quantum computational operations were executed on a very small number of qubits.[6] Both practical and theoretical research continues, and many national governments and military funding agencies support quantum computing research to develop quantum computers for both civilian and national security purposes, such as cryptanalysis.[7]

Large-scale quantum computers will be able to solve certain problems much quicker than any classical computer using the best currently known algorithms, like integer factorization using Shor's algorithm or the simulation of quantum many-body systems. There exist quantum algorithms, such as Simon's algorithm, which run faster than any possible probabilistic classical algorithm.[8] Given sufficient computational resources, however, a classical computer could be made to simulate any quantum algorithm; quantum computation does not violate the Church–Turing thesis.[9]

Quantum computer

From Wikipedia, the free encyclopedia



The Bloch sphere is a representation of a qubit, the fundamental building block of quantum computers.

A **quantum computer** is a computation system that makes direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data.[1] Quantum computers are different from digital computers based on transistors. Whereas digital computers require data to be encoded into binary digits (bits), each of which is always in one of two definite states (0 or 1), quantum computation uses qubits (quantum bits), which can be in superpositions of states. A theoretical model is the quantum Turing machine, also known as the universal quantum computer. Quantum computers share theoretical similarities with non-deterministic and probabilistic computers; one example is the ability to be in more than one state simultaneously. The field of quantum computing was first introduced by Yuri Manin in 1980[2] and Richard Feynman in 1982.[3][4] A quantum computer with spins as quantum bits was also formulated for use as a quantum space–time in 1968.[5]

As of 2014 quantum computing is still in its infancy but experiments have been carried out in which quantum computational operations were executed on a very small number of qubits.[6] Both practical and theoretical research continues, and many national governments and military funding agencies support quantum computing research to develop quantum computers for both civilian and national security purposes, such as cryptanalysis.[7]

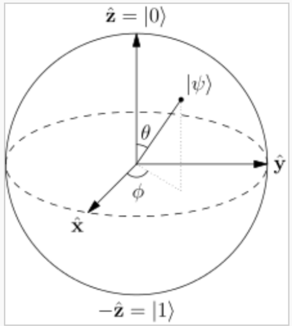
Large-scale quantum computers will be able to solve certain problems much quicker than any classical computer using the best currently known algorithms, like integer factorization using Shor's algorithm or the simulation of quantum many-body systems. There exist quantum algorithms, such as Simon's algorithm, which run faster than any possible probabilistic classical algorithm.[8] Given sufficient computational resources, however, a classical computer could be made to simulate any quantum algorithm; quantum computation does not violate the Church–Turing thesis.[9]

Both practical and theoretical research continues, and many national governments and military funding agencies support quantum computing research to develop quantum computers for both civilian and national security purposes, such as cryptanalysis.[7]

Quantum computer

From Wikipedia, the free encyclopedia

A **quantum computer** is a computation system that makes direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data.[1] Quantum computers are different from digital computers based on transistors. Whereas digital computers require data to be encoded into binary digits (bits), each of which is always in one of two definite states (0 or 1), quantum computation uses qubits (quantum bits), which can be in superpositions of states. A theoretical model is the quantum Turing machine, also known as the universal quantum computer. Quantum computers share theoretical similarities with non-deterministic and probabilistic computers; one example is the ability to be in more than one state simultaneously. The field of quantum computing was first introduced by Yuri Manin in 1980[2] and Richard Feynman in 1982.[3][4] A quantum computer with spins as quantum bits was also formulated for use as a quantum space–time in 1968.[5]



The Bloch sphere is a representation of a qubit, the fundamental building block of quantum computers.

As of 2014 quantum computing is still in its infancy but experiments have been carried out in which quantum computational operations were executed on a very small number of qubits.[6] Both practical and theoretical research continues, and many national governments and military funding agencies support quantum computing research to develop quantum computers for both civilian and national security purposes, such as cryptanalysis.[7]

Large-scale quantum computers will be able to solve certain problems much quicker than any classical computer using the best currently known algorithms, like integer factorization using Shor's algorithm or the simulation of quantum many-body systems. There exist quantum algorithms, such as Simon's algorithm, which run faster than any possible probabilistic classical algorithm.[8] Given sufficient computational resources, however, a classical computer could be made to simulate any quantum algorithm; quantum computation does not violate the Church–Turing thesis.[9]

Large-scale quantum computers will be able to solve certain problems much quicker than any classical computer using the best currently known algorithms, like integer factorization using Shor's algorithm

Shor’s algorithm and public-key crypto

- **What public-key systems it (supposedly) breaks**

  - Integer factorization (RSA)
  - Finite field discrete logarithm (DLP)
  - Elliptic curve discrete logarithm (ECDLP)

- **What public-key systems it (supposedly) doesn’t break**

  - Code-based cryptography (e.g. McEliece)
  - Hash-based cryptography
  - Multivariate cryptography (e.g. multiv. quadratic)
  - Lattice-based cryptography (e.g. LWE/R-LWE)

Shor's algorithm and public-key crypto

- **What public-key systems it (supposedly) breaks**

  - Integer factorization (RSA)
  - Finite field discrete logarithm (DLP)
  - Elliptic curve discrete logarithm (ECDLP)

- **What public-key systems it (supposedly) doesn't break**

  - Code-based cryptography (e.g. McEliece)
  - Hash-based cryptography
  - Multivariate cryptography (e.g. multiv. quadratic)
  - **Lattice-based cryptography (e.g. LWE/R-LWE)**

this talk

This work: R-LWE in TLS

- TLS (transport security layer) provides communication security for the internet
- All (non-pre-shared-key) ciphersuites currently offered in TLS will break if a large-scale quantum computer is built
- **This work:** build ciphersuites that (hopefully) won't

e.g. `RLWE-ECDSA-AES128-GCM-SHA256`

(openssl.org)

TLS v1.2 cipher suites

```
TLS_RSA_WITH_NULL_SHA256

TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_RSA_WITH_AES_128_CBC_SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA256
TLS_DH_RSA_WITH_AES_128_GCM_SHA256
TLS_DH_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_DSS_WITH_AES_128_CBC_SHA256
TLS_DH_DSS_WITH_AES_256_CBC_SHA256
TLS_DH_DSS_WITH_AES_128_GCM_SHA256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384

TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS_DH_anon_WITH_AES_128_CBC_SHA256
TLS_DH_anon_WITH_AES_256_CBC_SHA256
TLS_DH_anon_WITH_AES_128_GCM_SHA256
TLS_DH_anon_WITH_AES_256_GCM_SHA384
```

This work: R-LWE key agreement in TLS

- In this work, we start by looking at **post-quantum key agreement only**

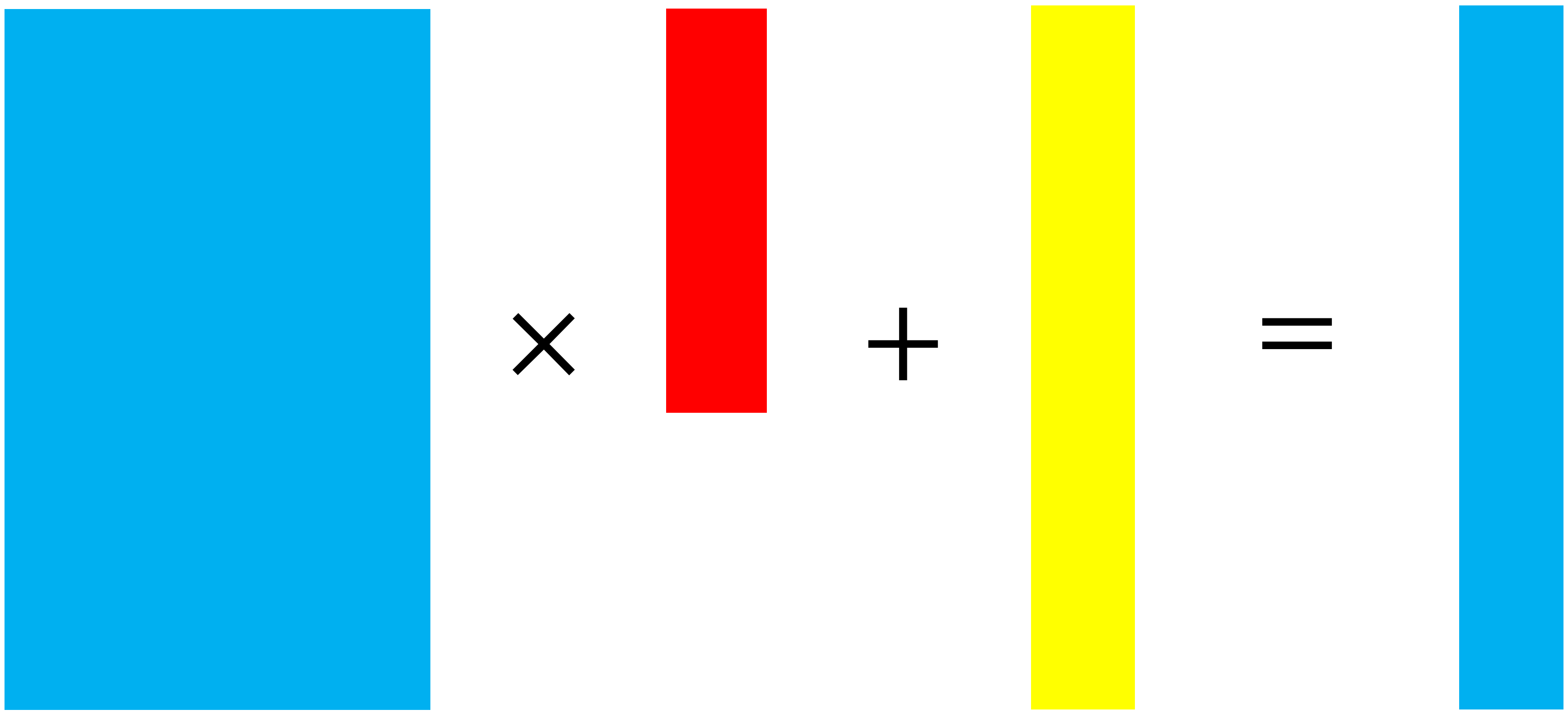
- **Assumption:** *large-scale quantum computers don't exist now, but what if we want to protect today's communications against tomorrow's adversary?*

- Signatures still done with traditional primitives RSA/ECDSA (we only need authentication to be secure now)

e.g. `RLWE-ECDSA-AES128-GCM-SHA256`

The learning with errors (LWE) problem

random	secret	small	ind. from random
$\mathbb{Z}_q^{m\times n}$	$\mathbb{Z}_q^{n\times 1}$	$\mathbb{Z}_q^{m\times 1}$	$\mathbb{Z}_q^{m\times 1}$



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

The learning with errors (LWE) problem

$$\begin{array}{ccccccc}
\textbf{random} & & \textbf{secret} & & \textbf{small} & & \textbf{ind. from random} \\
\boldsymbol{Z}_{13}^{7\times4} & & \boldsymbol{Z}_{13}^{4\times1} & & \boldsymbol{Z}_{13}^{7\times1} & & \boldsymbol{Z}_{13}^{7\times1} \\
\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix} & \times & \begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} & + & \begin{bmatrix} \ \\ \ \\ \ \\ \ \\ \ \\ \ \\ \ \end{bmatrix} & = & \begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix}
\end{array}$$

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

The learning with errors (LWE) problem

$$
\begin{array}{ccccccc}
\textbf{random} & & \textbf{secret} & & \textbf{small} & & \textbf{ind. from random} \\
\boldsymbol{Z}_{13}^{7\times 4} & & \boldsymbol{Z}_{13}^{4\times 1} & & \boldsymbol{Z}_{13}^{7\times 1} & & \boldsymbol{Z}_{13}^{7\times 1} \\
\begin{bmatrix} 4 & 1 & 11 & 10 \\ 5 & 5 & 9 & 5 \\ 3 & 9 & 0 & 10 \\ 1 & 3 & 3 & 2 \\ 12 & 7 & 3 & 4 \\ 6 & 5 & 11 & 4 \\ 3 & 3 & 5 & 0 \end{bmatrix}
& \times &
\begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix}
& + &
\begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \\ -1 \end{bmatrix}
& = &
\begin{bmatrix} 4 \\ 7 \\ 2 \\ 11 \\ 5 \\ 12 \\ 8 \end{bmatrix}
\end{array}
$$

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

Toy example versus real-world example

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

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

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

$$640\left\{\begin{array}{l}\overbrace{\begin{array}{ccccc} 2738 & 3842 & 3345 & 2979 & \cdots \\ 2896 & 595 & 3607 & & \\ 377 & 1575 & & & \\ 2760 & & & & \\ \vdots & & & & \end{array}}^{256}\end{array}\right.$$

$640 \times 256 \times 12 =$ 1966080 bits

= 245 kB !!

The learning with errors (LWE) problem

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

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

The ring learning with errors (R-LWE) problem

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

The ring learning with errors (R-LWE) problem

$$
\begin{array}{ccccccc}
\textbf{random} & & \textbf{secret} & & \textbf{small} & & \textbf{ind. from random} \\
\mathbb{Z}_{13}^{7\times 4} & & \mathbb{Z}_{13}^{4\times 1} & & \mathbb{Z}_{13}^{7\times 1} & & \mathbb{Z}_{13}^{7\times 1} \\
\begin{bmatrix} 4 & 1 & 11 & 10 \\ 3 & 4 & 1 & 11 \\ 2 & 3 & 4 & 1 \\ 12 & 2 & 3 & 4 \\ 12 & 7 & 3 & 4 \\ 9 & 12 & 7 & 3 \\ 10 & 9 & 12 & 7 \end{bmatrix} & \times & \begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix} & + & \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \\ -1 \end{bmatrix} & = & \begin{bmatrix} 4 \\ 3 \\ 4 \\ 12 \\ 5 \\ 12 \\ 11 \end{bmatrix}
\end{array}
$$

The ring learning with errors (R-LWE) problem

$$
\begin{array}{ccccccc}
\textbf{random} & & \textbf{secret} & & \textbf{small} & & \textbf{ind. from random} \\
\boldsymbol{Z}_{13}^{4\times 4} & & \boldsymbol{Z}_{13}^{4\times 1} & & \boldsymbol{Z}_{13}^{4\times 1} & & \boldsymbol{Z}_{13}^{4\times 1} \\
\begin{bmatrix} 4 & 1 & 11 & 10 \\ 3 & 4 & 1 & 11 \\ 2 & 3 & 4 & 1 \\ 12 & 2 & 3 & 4 \end{bmatrix} & \times & \begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix} & + & \begin{bmatrix} 0 \\ -1 \\ 1 \\ 1 \end{bmatrix} & = & \begin{bmatrix} 4 \\ 3 \\ 4 \\ 12 \end{bmatrix}
\end{array}
$$

LWE problem: given blue, find red

The ring learning with errors (R-LWE) problem

$$\boldsymbol{Z}_{13}^{4\times 4} \longrightarrow \boldsymbol{Z}_{13}[x]/(x^4+1)$$

$$\begin{array}{|c|c|c|c|l}
\hline
4 & 1 & 11 & 10 & \longrightarrow \quad 4+1x+11x^2+10x^3 \\
\hline
3 & 4 & 1 & 11 & \longrightarrow \quad = x\cdot(4+1x+11x^2+10x^3) \\
\hline
2 & 3 & 4 & 1 & \longrightarrow \quad = x^2\cdot(4+1x+11x^2+10x^3) \\
\hline
12 & 2 & 3 & 4 & \longrightarrow \quad = x^3\cdot(4+1x+11x^2+10x^3) \\
\hline
\end{array}$$

The ring learning with errors (R-LWE) problem

$$
\begin{array}{rl}
 & \mathbf{4 + 1x + 11x^2 + 10x^3} \\
\times & \mathbf{6 + 9x + 11x^2 + 11x^3} \\
+ & \mathbf{0 - 1x + 1x^2 + 1x^3} \\
\hline
 & \mathbf{10 + 5x + 10x^2 + 7x^3} \\
\hline
\end{array}
$$

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

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

The ring learning with errors (R-LWE) problem

$$
\begin{array}{rl}
 & 4 + 1x + 11x^2 + 10x^3 \\
\times & 1 + 0x - 1x^2 - 1x^3 \\
+ & 0 - 1x + 1x^2 + 1x^3 \\
\hline
 & 3 + 8x + 5x^2 + 6x^3 \\
\hline
\end{array}
\qquad \frac{\mathbb{Z}_{13}[x]}{\langle x^4 + 1 \rangle}
$$

R-LWE problem (small secrets): given **blue**, find (small!) **red**

The ring learning with errors (R-LWE) problem
(the 128-bit secure version)

$$\begin{array}{ll} & \mathbf{2792930407} + \cdots + \mathbf{2938465015}\boldsymbol{x}^{\mathbf{1023}} \\ \times & \mathbf{5} - \mathbf{3}\,\boldsymbol{x} \ldots + \mathbf{9}\boldsymbol{x}^{\mathbf{1022}} - \mathbf{1}\boldsymbol{x}^{\mathbf{1023}} \\ + & \mathbf{2} + \mathbf{4}\,\boldsymbol{x} \ldots - \mathbf{0}\boldsymbol{x}^{\mathbf{1022}} + \mathbf{6}\boldsymbol{x}^{\mathbf{1023}} \\ \hline & \mathbf{3159804584} + \cdots + \mathbf{1153769078}\boldsymbol{x}^{\mathbf{1023}} \\ \hline \end{array}$$

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

R-LWE problem: given **blue**, find (small!) **red**

R-LWE-DH: key agreement in $R_q = \boldsymbol{Z}_q[x]/\langle x^n + 1\rangle$

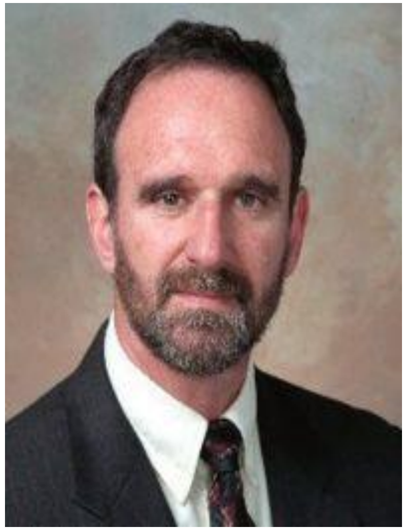
public: “big” a $\in R_q$

secret: “small” $e, s \in R_q$

$a \cdot s + e$ $\longrightarrow$

$\longleftarrow$ $a \cdot s' + e'$

secret: “small” $e', s' \in R_q$

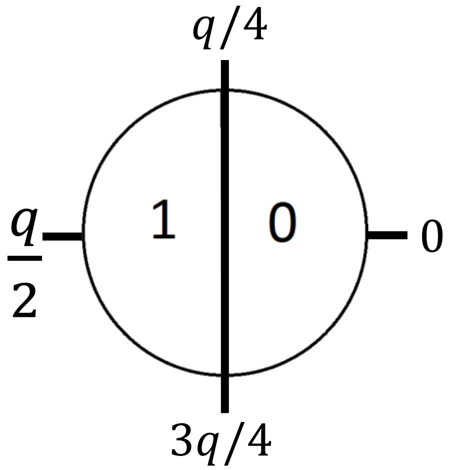


$\big(s \cdot (a \cdot s' + e')\big) \approx s \cdot a \cdot s'$

$\big(s' \cdot (a \cdot s + e)\big) \approx s \cdot a \cdot s'$

Approximate agreement mod q

the usual **ROUND** function



$$4079331841 + 1894732145 \cdot x + \cdots + 472608255 \cdot x^{1022} + 516748383 \cdot x^{1023}$$

$$\begin{array}{cccc} \approx & \approx & \approx & \approx \end{array}$$

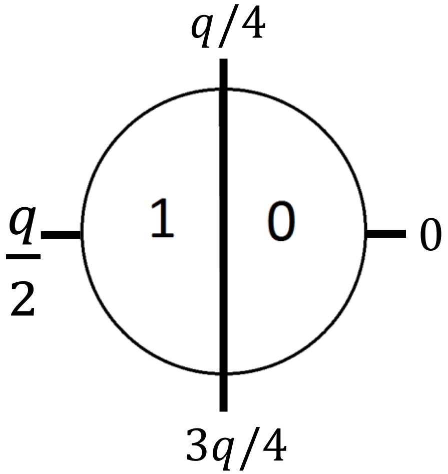
$$4079332556 + 1894733033 \cdot x + \cdots + 472607765 \cdot x^{1022} + 516748363 \cdot x^{1023}$$

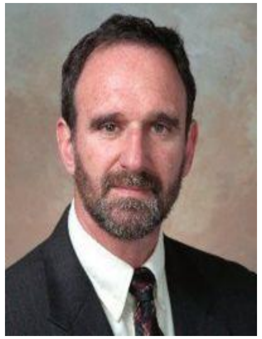
$$\begin{array}{cccc} \| & \| & \| & \| \end{array}$$

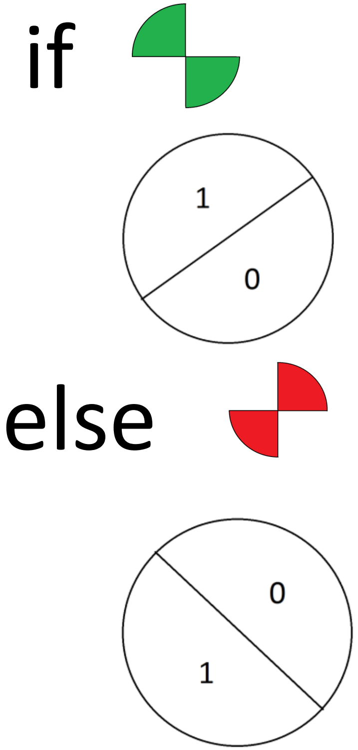
ROUND $\quad 0 \qquad 1 \qquad 0 \qquad 0$

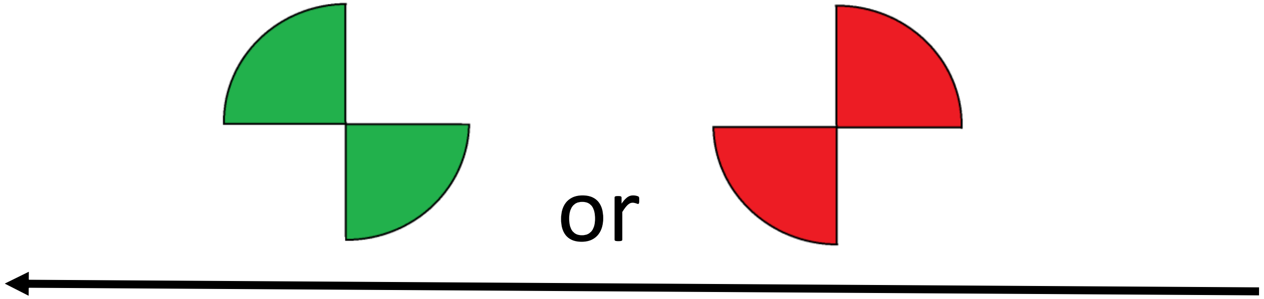
This will work most of the time (fails $\approx 1/2^{10}$), but we need **exact agreement**
i.e. what happens if one of the coefficients is in the **"danger zone(s)"**

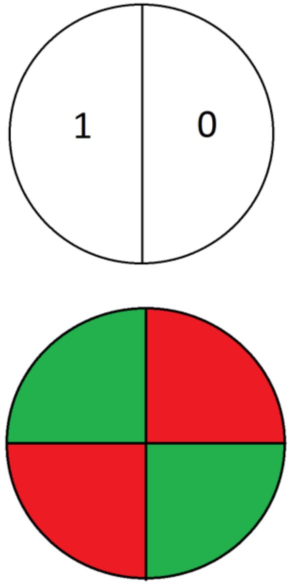
Making approximate agreement exact in $\boldsymbol{Z}_q$











(Peikert's reconciliation mechanisms: http://eprint.iacr.org/2014/070.pdf)
two values $u, v \in \boldsymbol{Z}_q$ will agree so long as $|u - v| < q/8$ (i.e. **always!**)

R-LWE-DH: exact key agreement

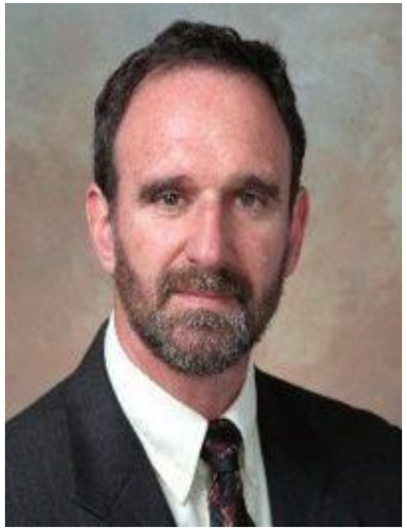
public: "big" a $\in R_q$

secret: "small" $e, s \in R_q$

$a \cdot s + e$

$a \cdot s' + e'$ and $\{\ ,\ \}^n \in \{0,1\}^n$

secret: "small" $e', s' \in R_q$



RECONCILE$(s \cdot (a \cdot s' + e'), \{\ ,\ \}^n)$ = ROUND$\left(s' \cdot (a \cdot s + e)\right)$

both parties now share $k \in \{0,1\}^n$

Security aspects

A secure key agreement protocol

- Prove that scheme is secure under "Exact DDH-like problem"
- Show that "Exact DDH-like problem" is hard if decision R-LWE problem is

Secure integration into the TLS

- Integrate R-LWE key agreement into the TLS protocol
- Use Jager *et al.* "Authenticated and confidential channel establishment" (ACCE) model (Crypto2012)
- Prove that "TLS-signed R-LWE is a secure ACCE"

Implementation aspect 1: polynomial arithmetic

- Polynomial multiplication in $R_q = \ \boldsymbol{Z}_q[x]/\langle x^{1024} + 1\rangle$ done with Nussbaumer's FFT ($2^l = r \cdot k$)

$$\frac{R[X]}{\langle X^{2^l} + 1\rangle} \cong \frac{\left(\frac{R[Z]}{\langle Z^r + 1\rangle}\right)[X]}{\langle X^k - Z\rangle}$$

- Rather than working modulo degree-1024 polynomial with coefficients in Z_q, work modulo:-
  - degree-256 polynomial whose coefficients are themselves polynomials modulo a degree-4 polynomial, or
  - degree-32 polynomials whose coefficients are polynomials modulo degree-8 polynomials whose coefficients are polynomials …

Implementation aspect 2: sampling discrete Gaussians

$$D_{\mathbf{Z},\sigma}(x) = \frac{1}{S}\, e^{-\frac{x^2}{2\sigma^2}} \quad \text{for } x \in \boldsymbol{Z}$$

(for us: $\sigma \approx 3.2$, $S = 8$)

- Security proofs require "small" elements to be within statistical distance 2^{-128} of *true* discrete Gaussian $D_{Z,\sigma}(x)$

- **Inversion sampling:** precompute table of cumulative probabilities (for us: 52 elements of 192-bits in size: ≈ 10,000 bits)

- Each coefficient requires six 192-bit integer comparisons (51 if "constant-time"), and there are 1024 coefficients!!!

The price of post-quantum paranoia, part I

Table 1: Average cycle count of standalone cryptographic operations (on client computer)

Operation	**Cycles**	
	constant-time	**non-constant-time**
sample $\xleftarrow{\$} \chi$	1 042 700	668 000
FFT multiplication	342 800	—
FFT addition	1 660	—
dbl$(\cdot)$ and crossrounding $\langle\cdot\rangle_{2q,2}$	23 500	21 300
rounding $\lfloor\cdot\rceil_{2q,2}$	5 500	3,700
reconciliation rec$(\cdot,\cdot)$	14 400	6 800

(Intel Core i5 (4570R) @ 2.7GHz)

The price of post-quantum paranoia, part II

Table 2: Average runtime in milliseconds of cryptographic operations using `openssl speed`

Operation	Client constant-time	Server constant-time	Client non-constant-time	Server non-constant-time
R-LWE key generation	0.9	1.7	0.6	1.3
R-LWE Bob shared secret	0.5	(1.1)	0.4	(0.9)
R-LWE Alice shared secret	(0.1)	0.4	(0.1)	0.4
Total R-LWE runtime	1.4	2.1	1.0	1.7
EC point multiplication, `nistp256`	0.4	0.7	—	—
Total ECDH runtime	0.8	1.4	—	—
RSA sign, 3072-bit key	(3.7)	8.8	—	—
RSA verify, 3072-bit key	0.1	(0.2)	—	—

Numbers in parentheses are reported for completeness, but do not contribute to the runtime in the client and server's role in the TLS protocol.

The price of post-quantum paranoia, part III

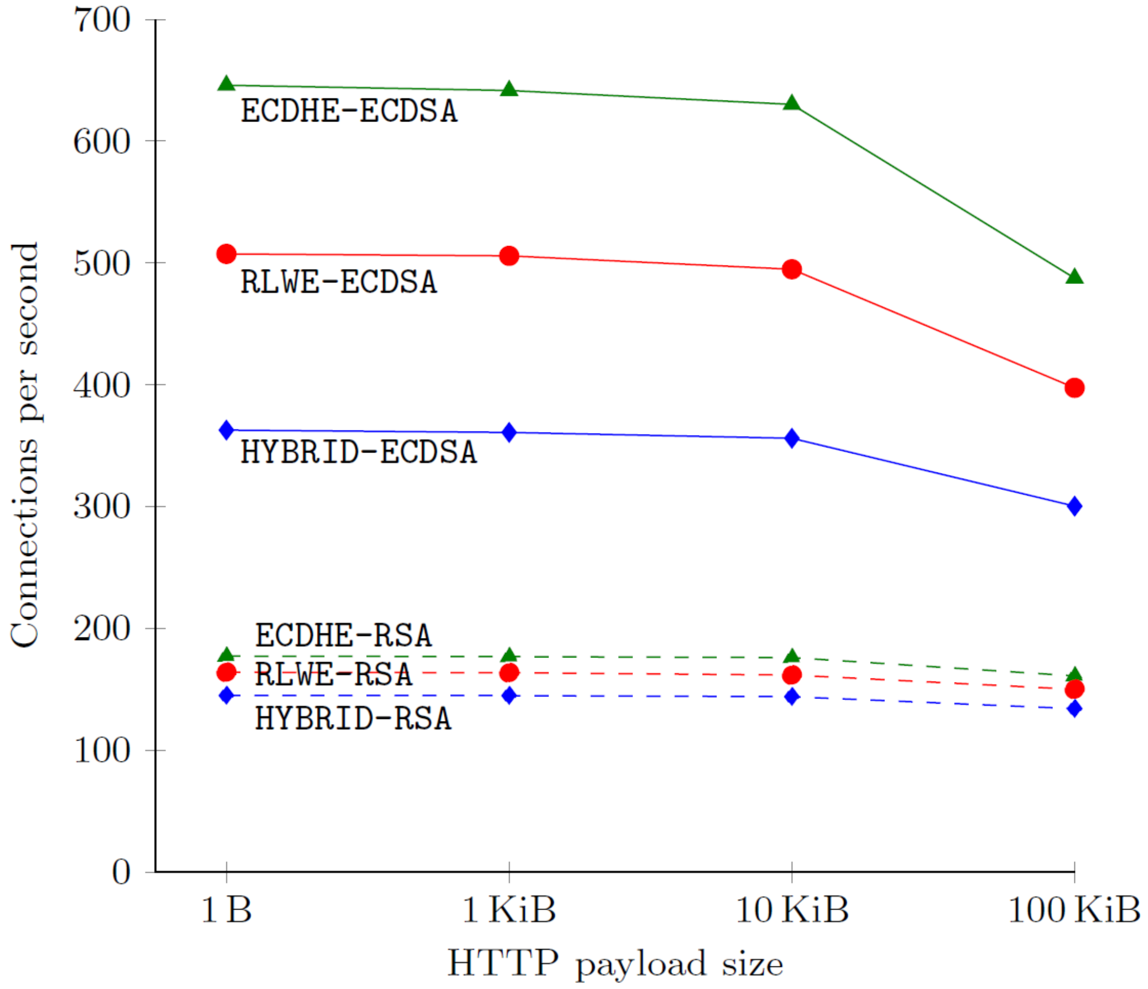


Table 3: Performance of HTTPS using Apache with OpenSSL

	ECDHE		RLWE		HYBRID	
	ECDSA	RSA	ECDSA	RSA	ECDSA	RSA
Connections / second						
— 1 B payload	645.9	177.4	507.5	164.2	362.9	145.1
— 1 KiB payload	641.6	177.0	505.9	163.8	361.0	145.0
— 10 KiB payload	630.2	176.2	494.9	161.9	356.2	144.1
— 100 KiB payload	487.6	161.2	397.6	150.2	300.5	134.3
Connection time (ms)	6.0	14.0	45.6	54.0	47.2	54.6
Handshake (bytes)	1 278	2 360	9 469	10 479	9 607	10 690

Summary and future work

- If you want to protect today's secrets against tomorrow's quantum adversary, use

RLWE-ECDSA-AES128-GCM-SHA256

in TLS for a small overhead

- **Future work, part II:** protecting tomorrow's secrets too!

RLWE-RLWE-AES128-GCM-SHA256

LWE-LWE-AES128-GCM-SHA256

????-????-AES128-GCM-SHA256

- **Future work, part I:** a tonne of unexplored optimizations (this is our first go)
  - e.g: we didn't do assembly/precomputation/parallelizing
  - e.g: alternative FFT's
  - e.g: much faster/compact sampling algorithms likely

The paper:
http://eprint.iacr.org/2014/599.pdf

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

C code integrated into OpenSSL:
https://github.com/dstebila/rlwekex