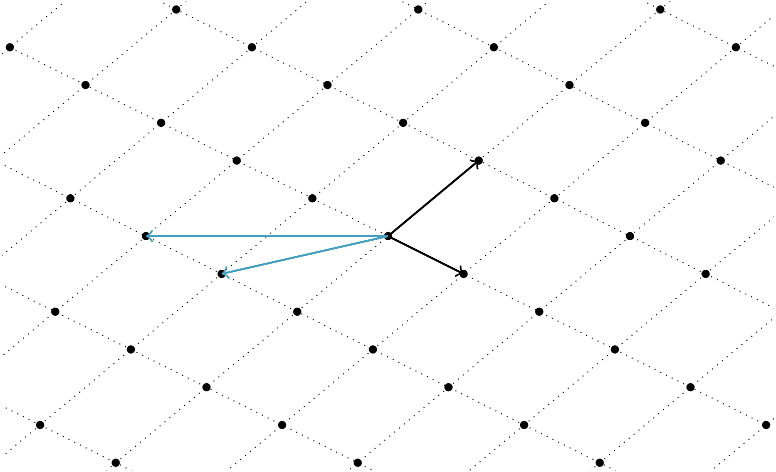
SRI International®

CRYSTALS (Crypto Suite for Algebraic Lattices) and Open Quantum Safe

Tancrède Lepoint
SRI International

Lattices



Lattices are represented by a basis. This basis is not unique.

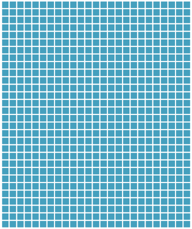
How did I construct this drawing?

Lattices

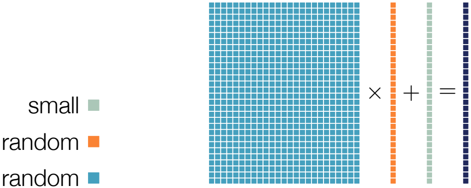
$$(x, y) \in \mathbb{Z}^2 \mapsto \begin{pmatrix} 1 & 1.2 \\ -0.5 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \in L$$

In cryptography:

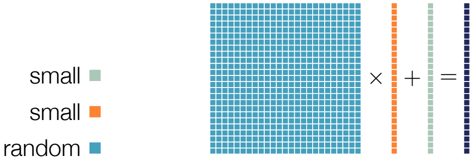
- matrices over $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ for an integer q
- in this talk, this is a lattice:



Hard Problem: Learning With Errors [Regev'05]



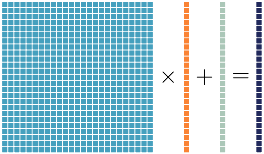
Hard Problem: Learning With Errors [Regev'05]

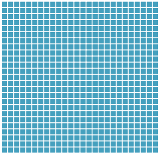
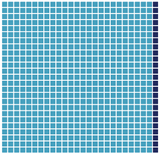


We can take the secret to be small, the matrix to be square [ACPS'09, LPR'10]

Hard Problem: Learning With Errors [Regev'05]

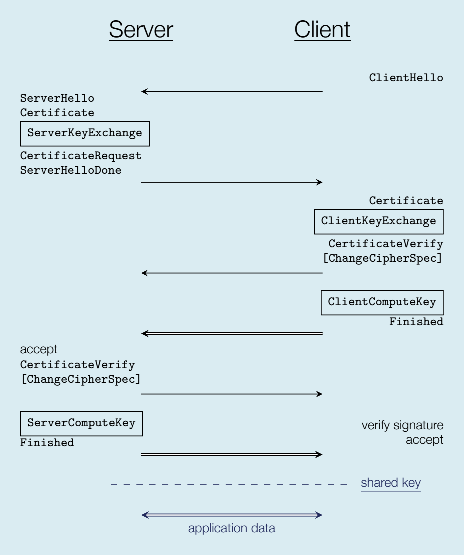
small
small
random



Decision LWE: Distinguish  and 

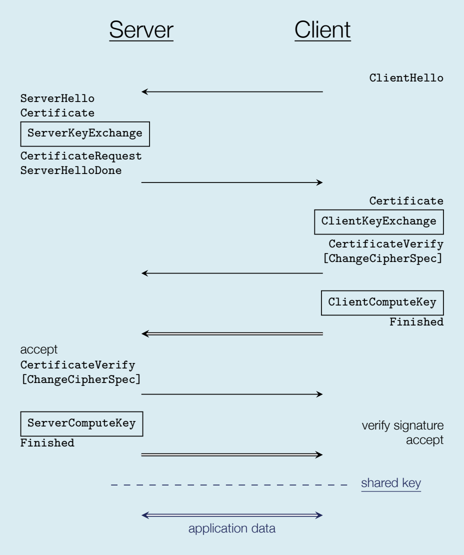
We can take the secret to be small, the matrix to be square [ACPS’09, LPR’10]

Key Encapsulation Mechanism in TLS



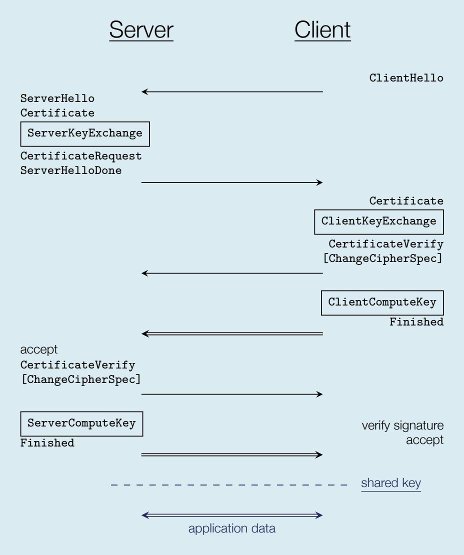
- Need replacement for **key exchange** (and **signature**)

Key Encapsulation Mechanism in TLS



- Need replacement for **key exchange** (and **signature**)
- We can use a **Key Encapsulation Mechanism**

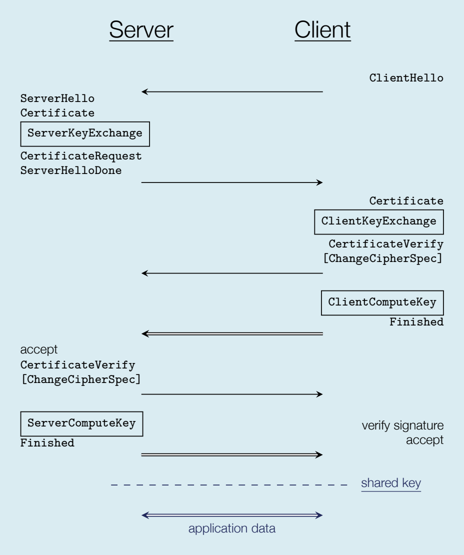
Key Encapsulation Mechanism in TLS



- Need replacement for **key exchange** (and **signature**)
- We can use a **Key Encapsulation Mechanism**

`ServerKeyExchange`: Generate a pk

Key Encapsulation Mechanism in TLS

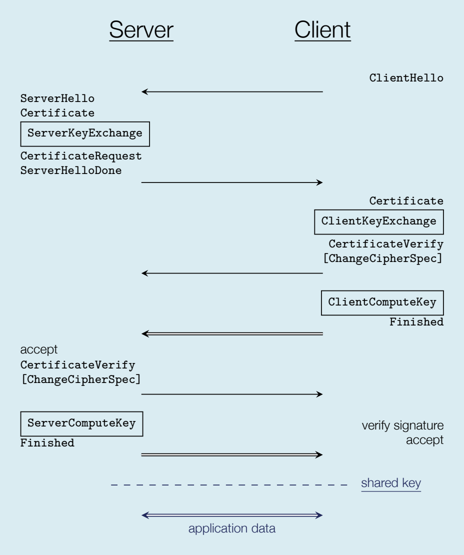


- Need replacement for **key exchange** (and **signature**)
- We can use a **Key Encapsulation Mechanism**

`ServerKeyExchange`: Generate a pk

`ClientKeyExchange`: Generate a session key $k = f(x)$ and encrypts x

Key Encapsulation Mechanism in TLS



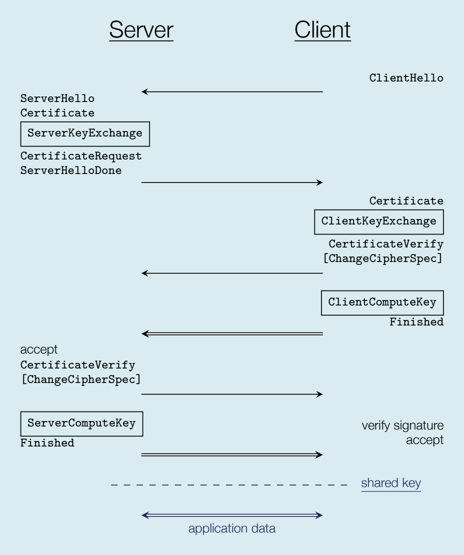
- Need replacement for **key exchange** (and **signature**)
- We can use a **Key Encapsulation Mechanism**

`ServerKeyExchange`: Generate a pk

`ClientKeyExchange`: Generate a session key $k = f(x)$ and encrypts x

`ServerComputeKey`: Decrypt to recover x and compute $k = f(x)$

Key Encapsulation Mechanism in TLS



- Need replacement for **key exchange** (and **signature**)
- We can use a **Key Encapsulation Mechanism**

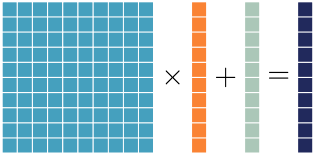
`ServerKeyExchange`: Generate a pk

`ClientKeyExchange`: Generate a session key $k = f(x)$ and encrypts x

`ServerComputeKey`: Decrypt to recover x and compute $k = f(x)$

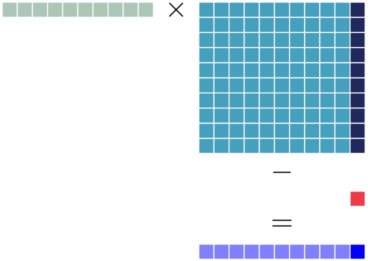
We want an encryption scheme :)

Regev's encryption scheme [Regev'05,GPV'08]

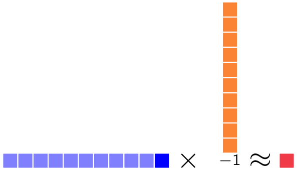


Key generation

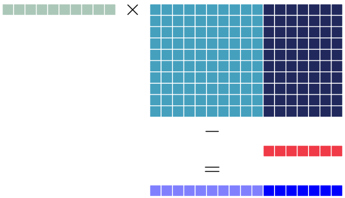
Encryption



Decryption



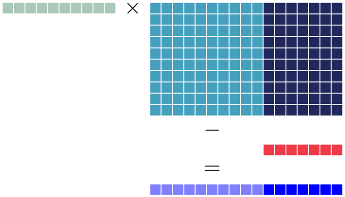
Using Regev's Scheme

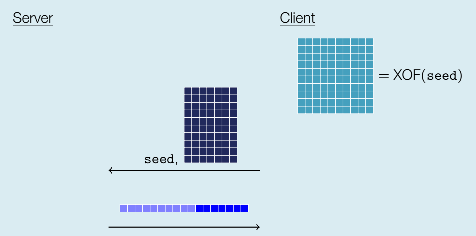


- Dimension of **A** $\approx 752 \times 752$
- Dimension of **B** $\approx 752 \times 256$
- Modulus $q = 2^{15} = 32768$

Using Regev's Scheme

In TLS:



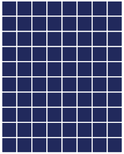


- Dimension of **A** $\approx 752 \times 752$
- Dimension of **B** $\approx 752 \times 256$
- Modulus $q = 2^{15} = 32768$

- $C \to S$:
  $752 \times 256 \times 15 = 2.8\text{Mb}$
- $S \to C$:
  $(752 + 256) \times 15 = 15\text{Kb}$

Reducing communication: Frodo (CCS'16)

- Public key: extract several bits per column (requires larger modulus) and compress



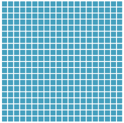
$\in \mathsf{trunc}_C(\mathbb{Z}_q^{256\times 8})$

(Comm.: 11,377 bytes)

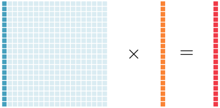
- Reconciliation instead of sending back ciphertext of key, set key $= \mathsf{LSB}_B(\mathbf{V})$ and send $B+1$-th bit of $\mathbf{V}$ (reconciliation)
  - Communication: 11,296 bytes
  - Idea from [Ding'12,Peikert'14]

Reducing communication with rings: NewHope (Usenix'16)

- Reconciliation (or not: NewHope-Simple)
- Use *ring* lattices [LM'06,PR'06], i.e.,

replace  by

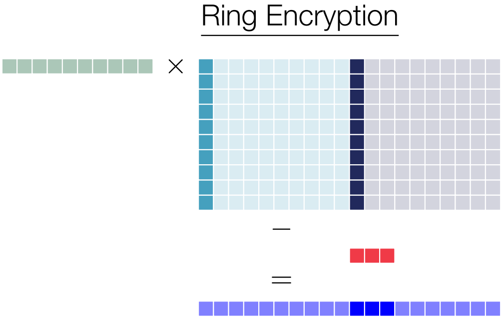
  - Columns: (anti-)cyclic rotations $\rightarrow$ need only to store n coefficients
  - Polynomial representation:



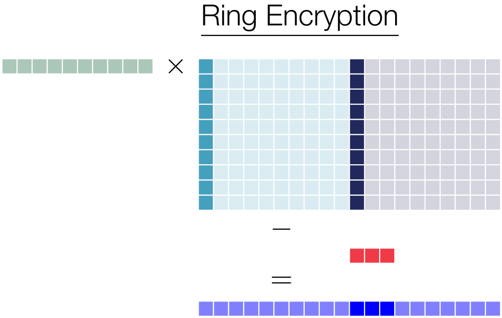
$\iff \mathbf{a}(x) \cdot \mathbf{s}(x) = \mathbf{p}(x)$ over $\mathbb{Z}_q[x]/(x^n + 1)$

- Communication: around 4KB (2kB each way)
- Successful experiment by Google from July to November 2016

Can we do better? Yes, because $\mathbf{1024} \gg \mathbf{256}$



Can we do better? Yes, because $\mathbf{1024} \gg \mathbf{256}$

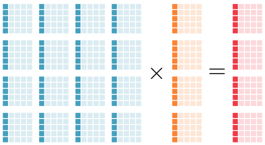


- In Frodo, we could select the **B** to only encrypt **K**
- In NewHope, we have many 0 coefficients
  - increases communication
  - we could use error-correcting codes to correct and reduce decryption error

CRYSTALS' Kyber (Real World Crypto 2017)

Joint work with Shi Bai, Joppe Bos, Léo Ducas, Eike Kiltz, Vadim Lyubashevsky, John M Schank, Peter Schwabe, and Damien Stehlé

Representation



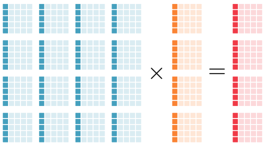
$\iff$

$\mathbf{A} \times \vec{s} = \vec{b}$ over $(\mathbb{Z}_q[x]/(x^{256}+1))^4$

CRYSTALS' Kyber (Real World Crypto 2017)

Joint work with Shi Bai, Joppe Bos, Léo Ducas, Eike Kiltz, Vadim Lyubashevsky, John M Schank, Peter Schwabe, and Damien Stehlé

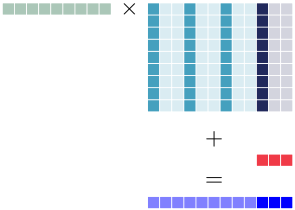
Representation



$\Longleftrightarrow$

$\mathbf{A} \times \vec{s} = \vec{b}$ over $(\mathbb{Z}_q[x]/(x^{256}+1))^4$

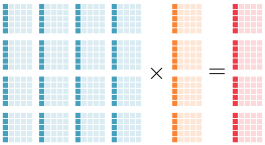
Module-Encryption



CRYSTALS' Kyber (Real World Crypto 2017)

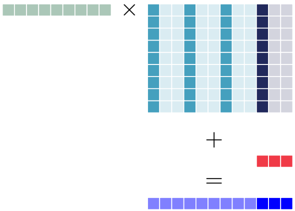
Joint work with Shi Bai, Joppe Bos, Léo Ducas, Eike Kiltz, Vadim Lyubashevsky, John M Schank, Peter Schwabe, and Damien Stehlé

Representation



$\Longleftrightarrow$

Module-Encryption



$\mathbf{A} \times \vec{s} = \vec{b}$ over $(\mathbb{Z}_q[x]/(x^{256}+1))^4$

- Communication: around 2KB (1KB each side)
- No reconciliation and better security (CCA scheme, long term keys, etc.)

Developers' aspects

Encapsulate(ρ, $\mathbf{t}$):

$m \leftarrow \{0,1\}^{256}$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim (\beta_\eta^{256k} \times \beta_\eta^{256k} \times \beta_\eta^{256}) := \mathrm{H}(m)$
$(\mathbf{u}, v) := \mathrm{Encrypt}\,((\rho, \mathbf{t}), m, (\mathbf{r}, \mathbf{e}_1, e_2))$
return $((\mathbf{u}, v), m)$

Decapsulate($(\rho, \mathbf{t}), (\mathbf{u}, v), \mathbf{s}$):

$m := \mathrm{Decrypt}((\mathbf{u}, v), \mathbf{s})$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim (\beta_\eta^{256k} \times \beta_\eta^{256k} \times \beta_\eta^{256}) := \mathrm{H}(m)$
$(\mathbf{u}', v') := \mathrm{Encrypt}((\rho, \mathbf{t}), m, (\mathbf{r}, \mathbf{e}_1, e_2))$
if $(\mathbf{u}, v) = (\mathbf{u}', v')$ **then**
  return m
end if

Fig. 3. CCA-KEM.

Building blocks:

- **Number Theoretic Transform** (dim 256)
- **Arithmetic modulo** $q = 7681 < 2^{13}$
- **"Noise" distribution:** draw 2η bits and compute $\sum_{i=1}^{\eta}(a_i - b_i)$
- **Working with module lattices:**

```
void polyvec_ntt(polyvec *r) {
  for(int i=0; i<3; i++) poly_ntt(&r->vec[i]);
}
```

- **Increasing security is easy:** essentially amounts to modify 3 in 4

Open Quantum Safe

`https://openquantumsafe.org`

`./openssl speed`

AWS `c4.large` (Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz)

Scheme	Server 0 (ms)	Client (ms)	Server 1 (ms)	Communication (bytes) $S \to C$	Communication (bytes) $C \to S$	Security (bits) Class.	Security (bits) PQ.
SIDH	15.84	35.14	14.97	564	564	192	128
McBits	69.92	0.04	0.15	311,736	109	157	157
LWE							
Frodo	0.91	1.33	0.16	11,377	11,296	144	130
Ring-LWE							
BCNS15	0.72	1.17	0.16	4,096	4,224	86	78
NewHope	0.05	0.08	0.02	1,824	2,048	281	255
NewHope-Simple				1,824	2,176		
Module-LWE							
Kyber	0.06	0.08	0.09	1,088	1,152	178	161

CRYSTALS: Wrap Up

CRYSTALS: *Cryptographic Suite for Algebraic Lattices*

- KYBER: CCA-KEM
  - Can be used for key exchange, and in KEM+DEM
  - Similar to NewHope (i.e., no impediment to use it)
  - Better communication and better security than NewHope
  - Nearly as efficient as NewHope
  - Easy to implement: arithmetic mod 7681, 256-NTT, sum of bits. No reconciliation or Gaussian sampling.
- DILITHIUM: Digital Signature
  - Same building blocks than Kyber with module lattices
    - easiness of implementation, reusability
  - Uniform error distributions
  - Fiat-Shamir paradigm

SRI International®

Thank You

https://github.com/pq-crystals

Headquarters
333 Ravenswood Avenue
Menlo Park, CA 94025
+1.650.859.2000

New York Office
60 E 42nd St, Suite 1650
New York, NY 10165
+1.646.693.0930

Washington D.C. Office
1100 Wilson Boulevard, Suite 2800
Arlington, VA 22209
+1.703.524.2053

Additional U.S. and international locations

www.sri.com