Standardizing Lattice Cryptography … and Beyond

Vadim Lyubashevsky

IBM Research – Zurich

Why Lattice Cryptography

- One of the oldest and most (the most?) efficient quantum-resilient alternatives for “basic primitives”
  - Public key encryption
  - Digital signatures

- Many “advanced” primitives can be based on these hardness assumptions

CRYSTALS

Cryptographic Suite for Algebraic Lattices

Joppe Bos Leo Ducas

Eike Kiltz Tancrede Lepoint

Vadim Lyubashevsky John Schanck

Peter Schwabe Gregor Seiler Damien Stehle

CRYSTALS: KYBER

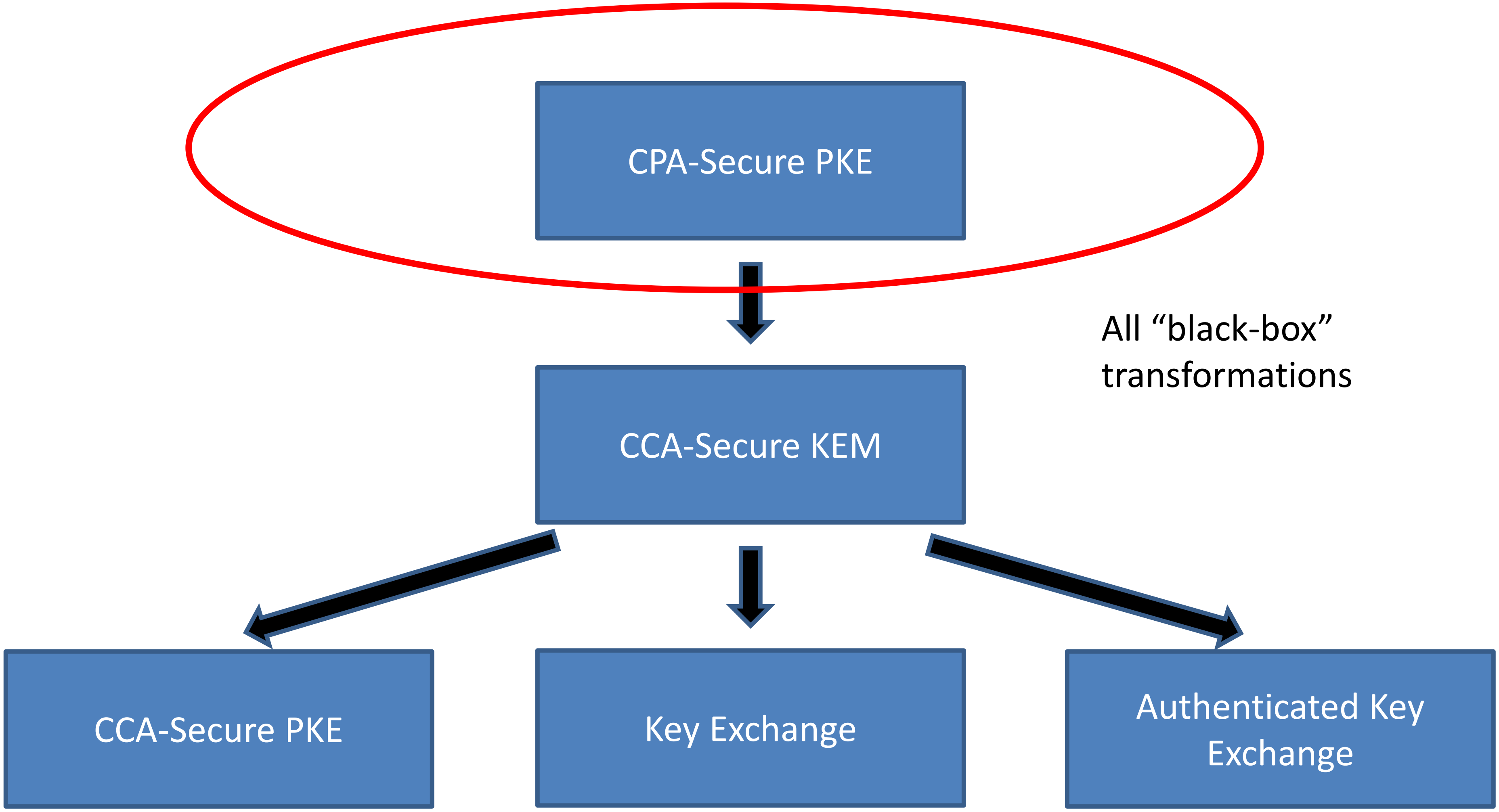
CCA KEM (AND ENCRYPTION)

Design Philosophies

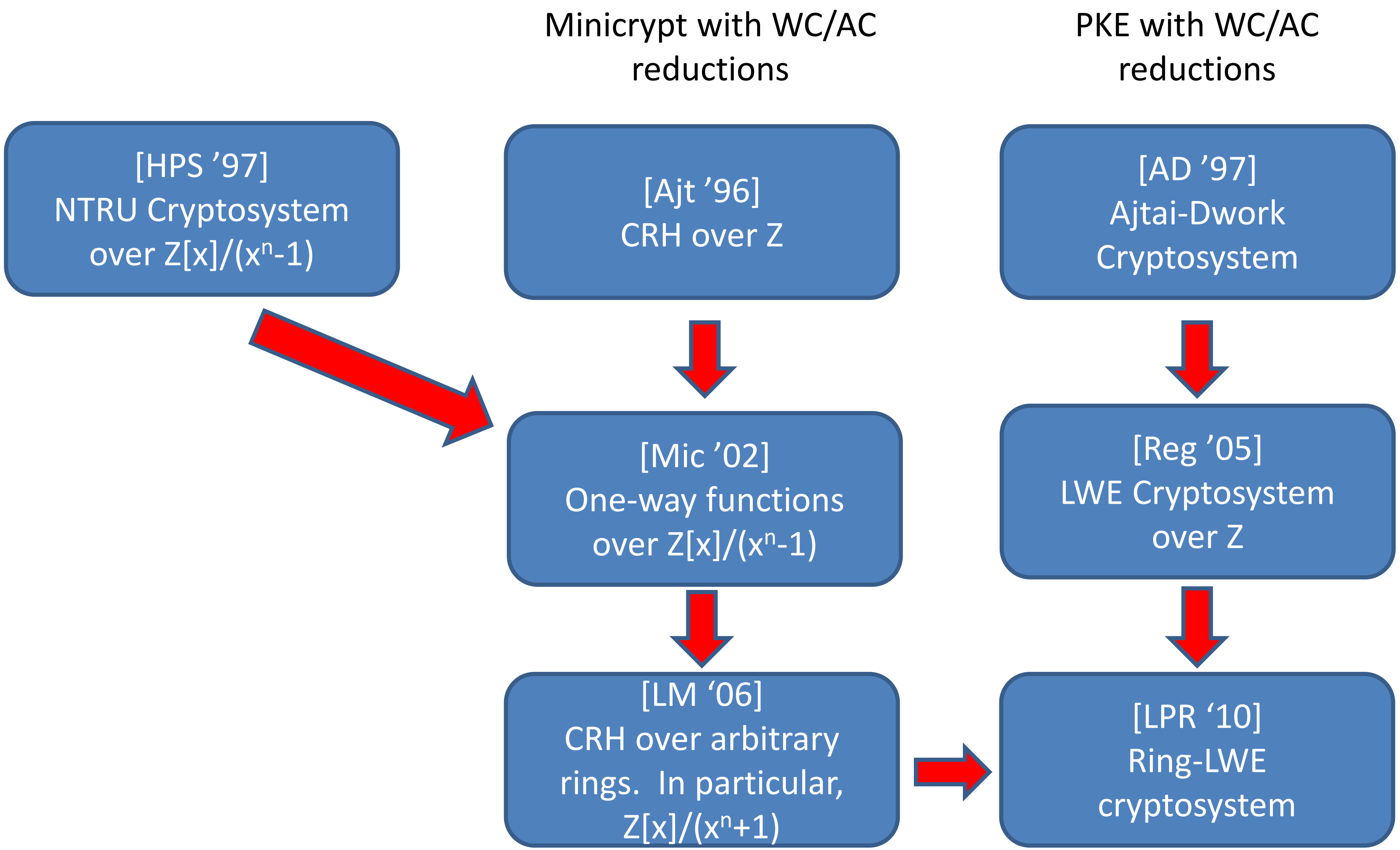
- CCA only
  - The primitives are already very fast; no need to set speed records

- Make adjusting security levels simple – always operate over the ring $Z_q[X]/(X^{256}+1)$ for $q=2^{13}-2^9+1$
  - If you care about post-quantum security, you can start implementing/optimizing/using now
  - Scheme can be easily adjusted once more exact cryptanalysis is agreed upon

Key Exchange / CCA – Encryption/ Authenticated Key Exchange



PKE Development



Giving Credit

- Hoffstein, Pipher, Silverman
  - Cryptosystem Using Polynomial Rings '97
- Ajtai, Dwork
  - General Lattice Cryptosystem '97
- Alekhnovich
  - LPN-Based Cryptosystem '03
- Regev
  - LWE Cryptosystem '05
- Lyubashevsky, Peikert, Regev
  - Practical (Ring)-LWE Cryptosystem '10

Giving Credit

- Hoffstein, Pipher, Silverman
  - Cryptosystem Using Polynomial Rings ‘97
- Ajtai, Dwork
  - General Lattice Cryptosystem ‘97
- Alekhnovich
  - LPN-Based Cryptosystem ‘03
- Regev
  - LWE Cryptosystem ‘05
- Lyubashevsky, Peikert, Regev
  - Practical (Ring)-LWE Cryptosystem ‘10

Hard Apples

- Hoffstein, Pipher, Silverman
  – Cryptosystem Using Polynomial Rings '97
- Ajtai, Dwork
  – General Lattice Cryptosystem '97
- Alekhnovich
  – LPN-Based Cryptosystem '03
- Regev
  – LWE Cryptosystem '05
- Lyubashevsky, Peikert, Regev
  – Practical (Ring)-LWE Cryptosystem '10

Hard Apples

- Hoffstein, Pipher, Silverman
  - Cryptosystem Using Polynomial Rings '97
- Ajtai, Dwork
  - General Lattice Cryptosystem '97
- Alekhnovich
  - LPN-Based Cryptosystem '03
- Regev
  - LWE Cryptosystem '05
- Lyubashevsky, Peikert, Regev
  - ~~Practical (Ring)-LWE~~ Cryptosystem '10

    Hard Apples

The Polynomial Ring $Z_q[x]/(x^d+1)$

$R = Z_q[x]/(x^d+1)$ is a polynomial ring with

- Addition mod q
- Polynomial multiplication mod q and x^d+1

Each element of R consists of d elements in Z_q

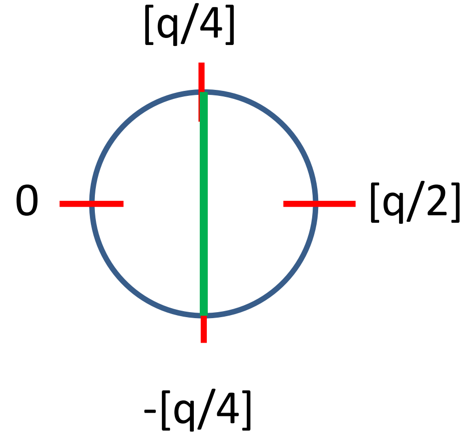
In R:

- small+small = small
- small*small = small

(Note: If d=1, then $R=Z_q^*$)

Rounding Function

$Round_1(w)$



$Round_k(w)$ = " Round w to the nearest [q/2] "

Hard Apples Encryption [LPR '10]

KeyGen:

$A \leftarrow R^{n \times n}$

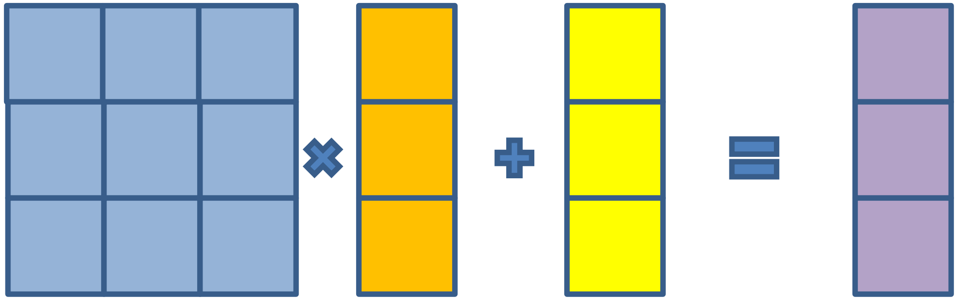
$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A, t)

sk: s

Hard Apples Encryption [LPR '10]



Public Key / Secret Key Generation

Hard Apples Encryption [LPR '10]

KeyGen:

$A \leftarrow R^{n \times n}$

$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A,t)

sk: s

Encrypt(μ):

$r', e' \leftarrow \psi^n$

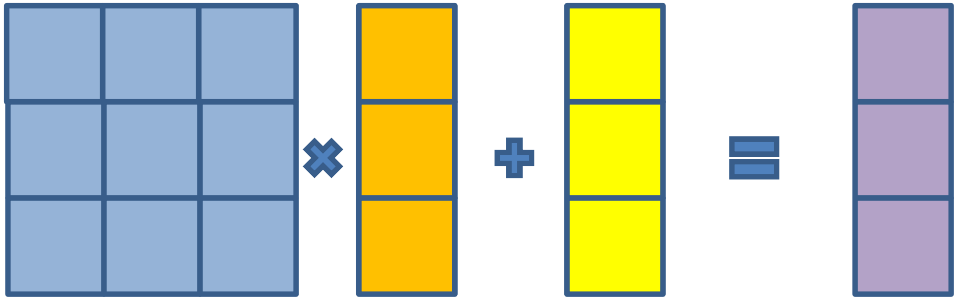
$f \leftarrow \psi$

$u' := r'A + e'$

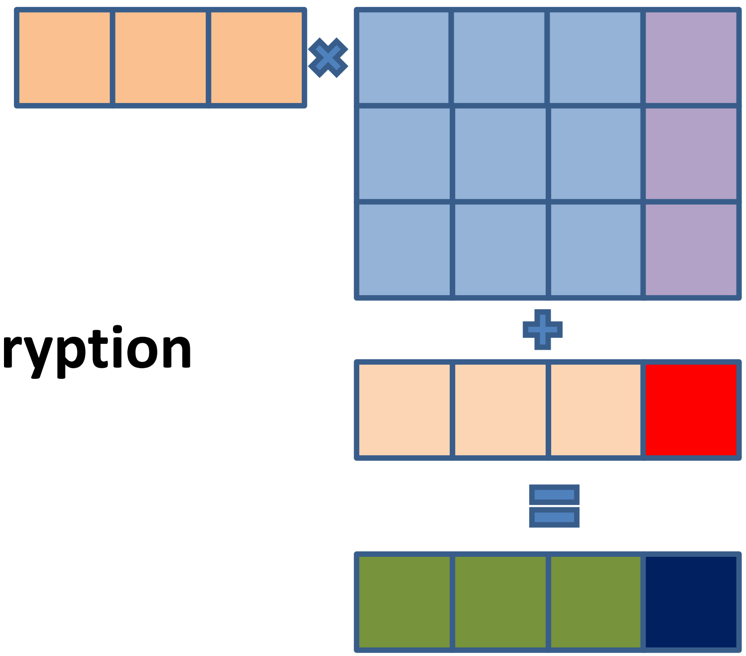
$v := r't + f + [q/2]\mu$

ciphertext: (u',v)

Hard Apples Encryption [LPR '10]



Public Key / Secret Key Generation



Encryption

Hard Apples Encryption [LPR '10]

KeyGen:

$A \leftarrow R^{n \times n}$

$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A,t)

sk: s

Encrypt(μ):

$r', e' \leftarrow \psi^n$

$f \leftarrow \psi$

$u' := r'A + e'$

$v := r't + f + [q/2]\mu$

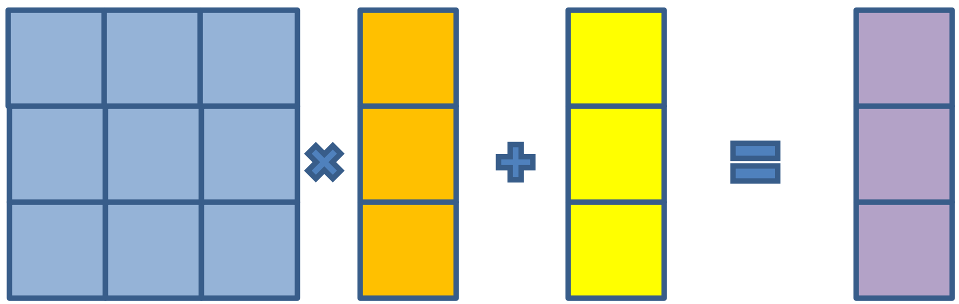
ciphertext: (u',v)

Decrypt(u',v):

$w := v - u's$

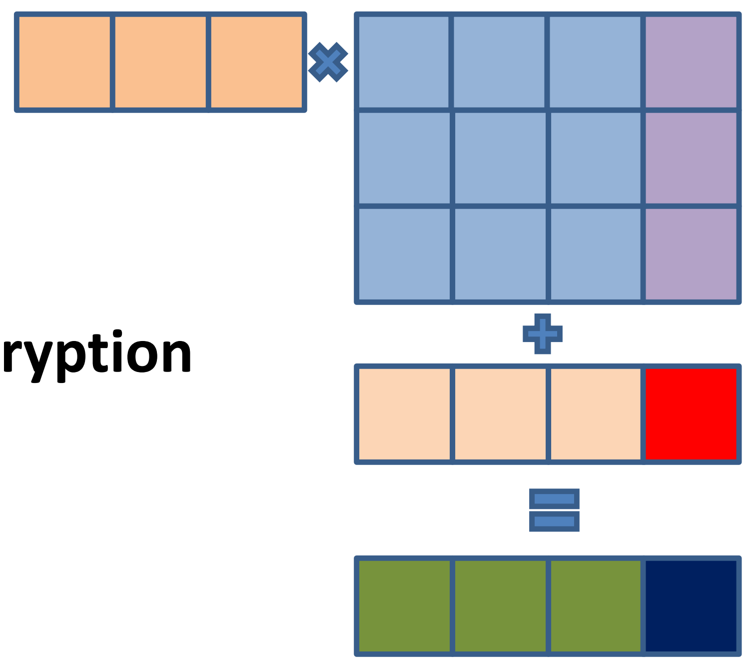
$$\mu := \frac{\mathrm{Round}_1(w)}{[q/2]}$$

Hard Apples Encryption [LPR '10]

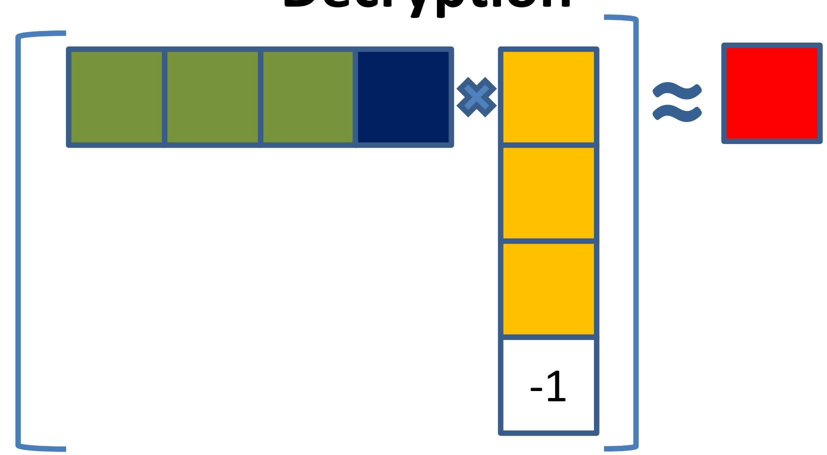


Public Key / Secret Key Generation

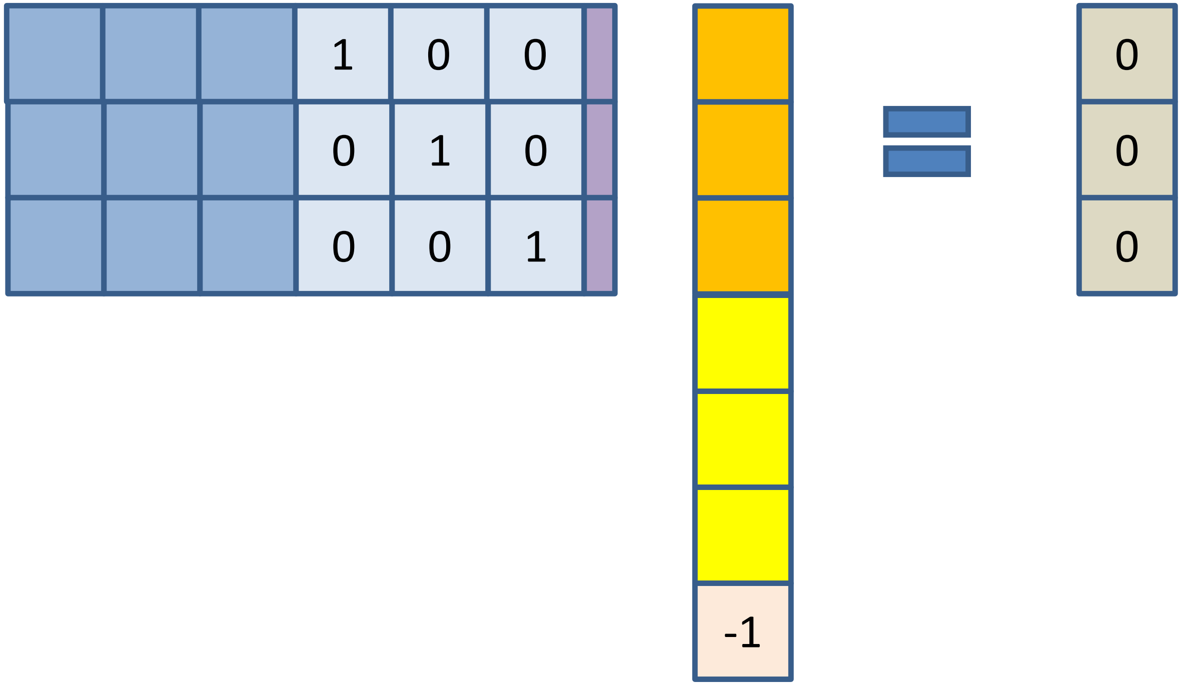
Encryption



Decryption



Practical Security



Best attack is finding the shortest vector in a lattice of dimension 2nd+1

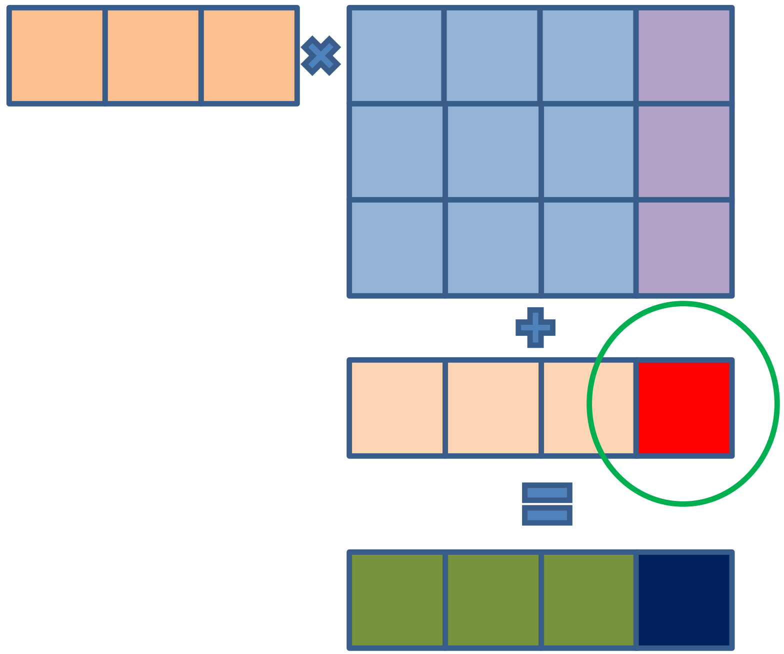
Relation to LWE and Ring-LWE

- In LWE, d=1
  - Security completely dependent on n

- In Ring-LWE, n=1
  - Security completely dependent on d

Message Space Size

Encryption



message = 1 element in R with 0/1 coefficients

d coefficients

Larger d → Larger message

But 256-bit messages are enough → Can set d=256

Hard Apples vs. NTRU

Public key size, ciphertext size, encryption, decryption, all approximately the same

NTRU key generation ≈ 10x slower

Main disadvantage of NTRU: Geometric structure of the NTRU lattice [KF '17]

Breaks NTRU for large q, small ψ

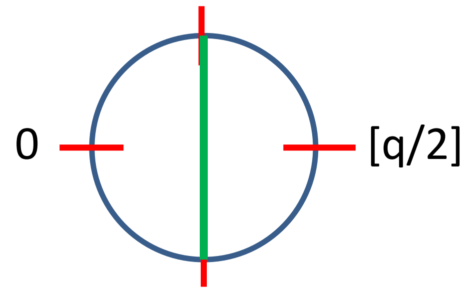
Is NTRU Broken?

- No. For a small modulus as used in encryption, it's still secure.
- No attack in the past 20 years actually threatened NTRU or Hard Apples
  - (Even the recent incorrect quantum algorithm of Eldar and Shor didn't break these schemes)
- But … advanced schemes (like FHE) where q must be large will be broken if based on NTRU
- Geometric structure could be exploited further

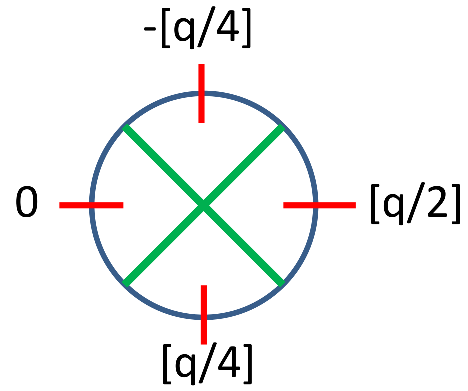
SIMPLE EFFICIENCY IMPROVEMENTS

Rounding Function

$\mathrm{Round}_1(w)$



$\mathrm{Round}_2(w)$



$\mathrm{Round}_k(w)$ = " Round w to the nearest $q/2^k$ "

$$|w - \mathrm{Round}_k(w)| < q/2^{k+1}$$

Hard Apples Encryption [LPR '10]

KeyGen:

$A \leftarrow R^{n \times n}$

$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A,t)

sk: s

Encrypt(μ):

$r', e' \leftarrow \psi^n$

$f \leftarrow \psi$

$u' := r'A + e'$

$v := r't + f + \lceil q/2 \rceil \mu$

ciphertext: (u',v)

Decrypt(u',v):

$w := v - u's$

$$\mu := \frac{\mathrm{Round}_1(w)}{\lceil q/2 \rceil}$$

$$w := v - u's = r'e - e's + f + \lceil q/2 \rceil \mu$$

Each coefficient of $|r'e - e's + f|$ should be less than q/4

Hard Apples Encryption [LPR '10]

KeyGen:

$A \leftarrow R^{n \times n}$

$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A, t)

sk: s

Encrypt(μ):

$r', e' \leftarrow \psi^n$

$f \leftarrow \psi$

$u' := r'A + e'$

$v := \mathrm{Round}_k(r't + f + [q/2]\mu)$

ciphertext: (u', v)

Decrypt(u',v):

$w := v - u's$

$\mu := \dfrac{\mathrm{Round}_1(w)}{[q/2]}$

$$w := v - u's = r'e - e's + f + [q/2]\mu + \boldsymbol{\varepsilon_v}$$

Each coefficient of $|\varepsilon_v|$ is at most $q/2^{k+1}$

Each coefficient of $|r'e - e's + f|$ should be less than $q/4 - q/2^{k+1}$

INTERLUDE: COMPARISON WITH "RECONCILIATION-BASED" KEM

(Preview: This is not better than PKE)

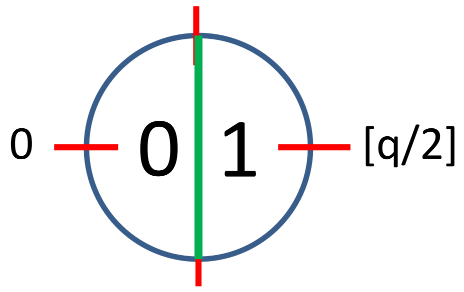
Reconciliation

Player 1 gets a random value x mod q
Player 2 gets some value y such that |x-y mod q|<ε

Player 1 and 2 want to secretly agree on 1 bit.
This is not possible without additional communication

Upon receiving x, player 1 sends a “hint” to player 2 such that:

1. x and y can agree on a bit
2. anyone who only sees the hint cannot guess the bit



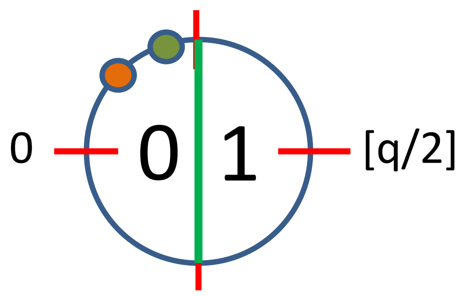
Reconciliation

Player 1 gets a random value x mod q
Player 2 gets some value y such that |x-y mod q|<ε

Player 1 and 2 want to secretly agree on 1 bit.
This is not possible without additional communication

Upon receiving x, player 1 sends a "hint" to player 2 such that:

1. x and y can agree on a bit
2. anyone who only sees the hint cannot guess the bit



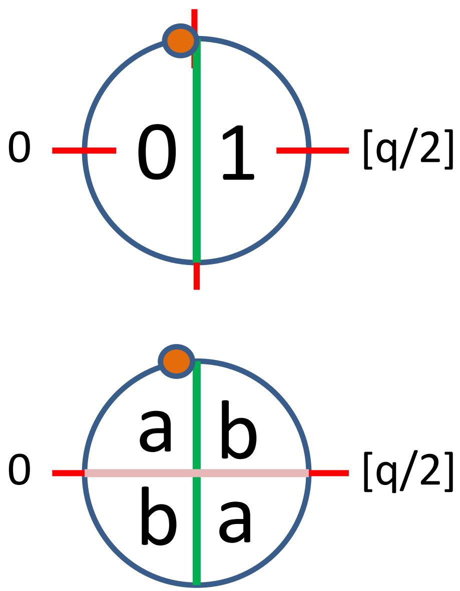
Reconciliation

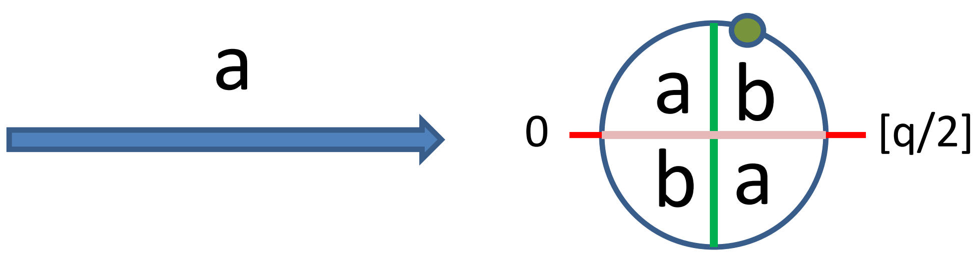
Player 1 gets a random value x mod q
Player 2 gets some value y such that |x-y mod q|<ε

Player 1 and 2 want to secretly agree on 1 bit.
This is not possible without additional communication

Upon receiving x, player 1 sends a “hint” to player 2 such that:

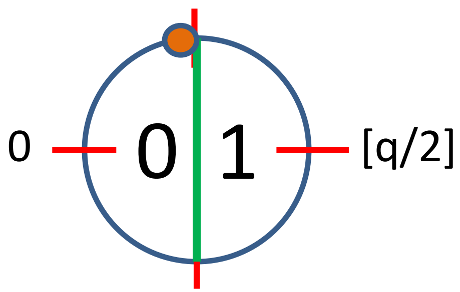
1. x and y can agree on a bit
2. anyone who only sees the hint cannot guess the bit

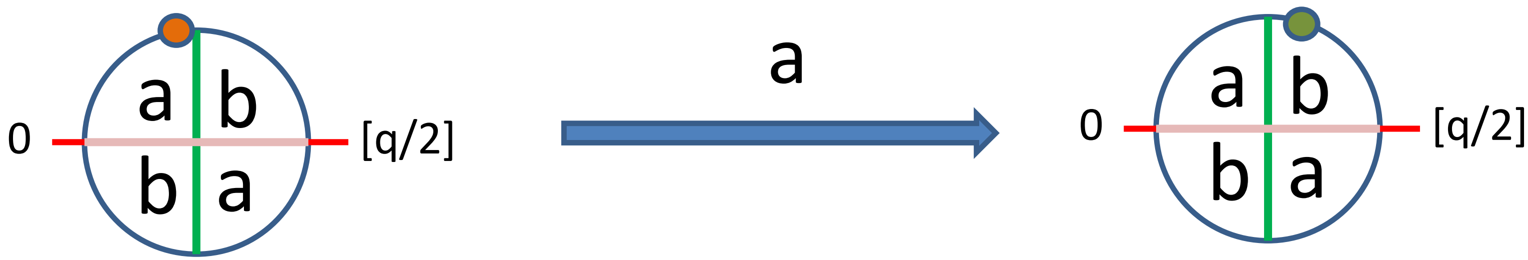




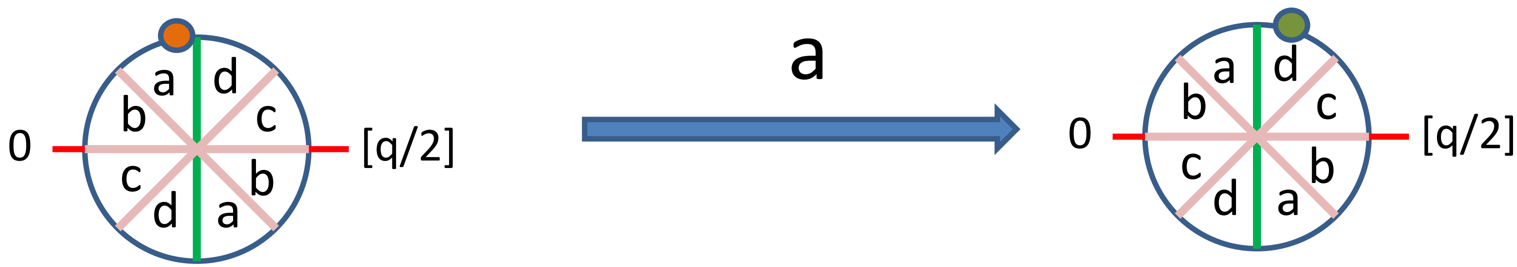
If ε < q/8, then Player 2 will know which half x is in

Allowing for Larger ε





If ε < q/8, then Player 2 will know which half x is in



If ε < 3q/16, then Player 2 will know which half x is in

k "hint bits" → if $\varepsilon < q/4 - q/2^{k+2}$, then Player 2 will know which half x is in

KEM Based on Reconciliation [D '12, P'14]

KeyGen:

$A \leftarrow R^{n \times n}$

$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A,t')

sk: s'

Encapsulate():

$r', e' \leftarrow \psi^n$

$f \leftarrow \psi$

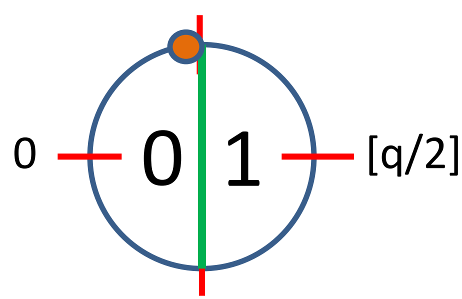
$u' := r'A + e'$

$v := \mathrm{HintBits}_k(r't + f)$

$= \mathrm{HintBits}_k(r'As + r'e + f)$

ciphertext: (u',v)

$\lambda := \mathrm{Round}_1(v)$

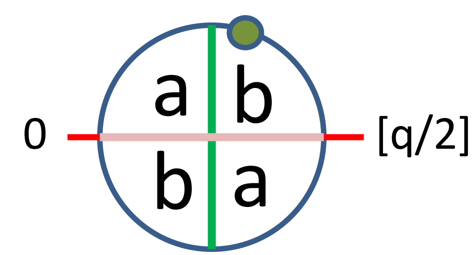


Decapsulate(u',v):

$w := u's$

$(= r'As + e's)$

$\lambda := \mathrm{Reconc}(w, v)$



Comparing Encryption and Reconciliation KEM

<u>Public Key Encryption</u>

To encrypt 256-bit message:

$nd \log q + dk + 256$ bits

<u>KEM</u>

To share 256-bit key:

$nd \log q + dk$ bits

In practice, the KEM is about 256 bits ≈ 3% shorter, but …

both the Encryption scheme and KEM are only passively-secure

Start with KEM or PKE?

For our application, there is **no difference**

PKE is just simpler and more direct

Maybe one can go from KEM to something useful and save a little bit … perhaps with error correction, but I'm not sure

But it's definitely **not** as stated in [P '14]:

naïve

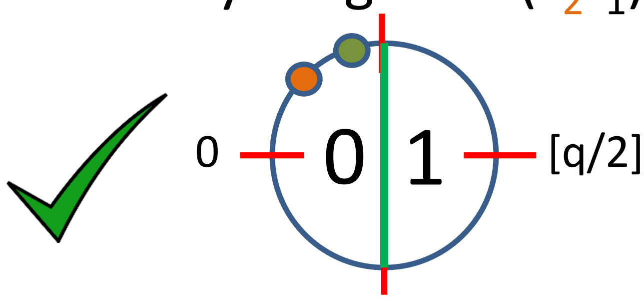
"As compared with the ~~previous most efficient~~ ring-LWE cryptosystems and KEMs, the new reconciliation mechanism reduces the ciphertext length by nearly a factor of two, because it replaces one of the ciphertext's two R_q elements with an R_2 element."

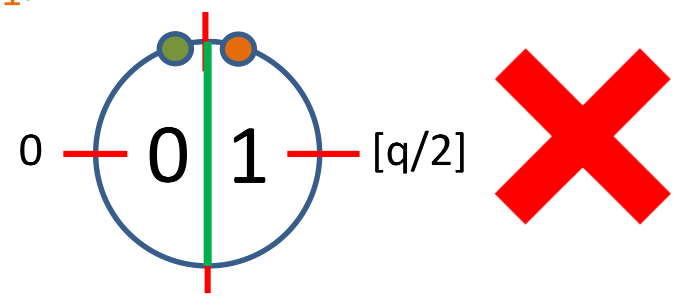
Interlude: Non-Interactive "Diffie-Hellman"-like Key Exchange

Common randomness **A**

Player 1 Public Key: $\mathbf{t}_1 = \mathbf{A}\mathbf{s}_1 + \mathbf{e}_1$
Player 2 Public Key: $\mathbf{t}_2 = \mathbf{s}_2\mathbf{A} + \mathbf{e}_2$

Joint key: HighBits (s_2t_1) = HighBits(t_2s_1)





Error happens with probability ≈ $|\mathbf{s}_2\mathbf{e}_1| / q \approx |\mathbf{e}_2\mathbf{s}_1| / q$
PK sizes of (probably) more than 40 - 50 KB
Double that if $\mathbf{s}_1\mathbf{A}$ is not $\mathbf{A}\mathbf{s}_1$
using Ring-LWE is twice as efficient as using Module-LWE

Varieties of Hard Apples

- Use LWE instead of Ring-LWE / Module-LWE (Frodo)

  Pros: No algebraic structure to try and exploit in attacks

  Cons: 10x slower, 10x larger public key, 10x larger ciphertext (when trying to minimize size of public key + ciphertext)

- Use Ring-LWE (i.e. set n=1) instead of Module-LWE (with flexible n) (New Hope Light)

  Pros: A little faster

  Cons: Less flexible (if the degree is a power of 2), smaller n could affect practical security

- Use rounding instead of adding random errors (Lizard,NTRU-Prime)

  Pros: A little faster

  Cons: Unclear if deterministic noise leads to new attacks (a very aggressive version of LWR)

- Use a ring Z[X]/(f(x)) for a different f(x) (NTRU-Prime)

  Pros: Algebraic attacks could be less obvious than for $f(x)=x^d+1$

  Cons: A little slower, slightly larger "expansion factor" , no algebraic structure that's useful for some advanced applications

FURTHER PKE EFFICIENCY IMPROVEMENTS

Hard Apples Encryption [LPR '10]

KeyGen:

$A \leftarrow R^{n \times n}$

$s,e \leftarrow \psi^n$

$t := \text{Round}_\alpha(As+e)$

pk: (A,t)

sk: s

Encrypt(μ):

$r',e' \leftarrow \psi^n$

$f \leftarrow \psi$

$u' := \text{Round}_\alpha(r'A+e')$

$v := \text{Round}_k(r't+f+[q/2]\mu)$

ciphertext: (u',v)

Decrypt(u',v):

$w := v-u's$

$\mu := \frac{\text{Round}_1(w)}{[q/2]}$

$$w := v-u's = r'e - e's + f + [q/2]\mu + \varepsilon_v + r'\varepsilon_t + \varepsilon_u s$$

Set the size for security

Larger ε → smaller pk / ciphertext …
but larger decryption error
Need to manually optimize

Added "Benefit" of Rounding

KeyGen:

$A \leftarrow R^{n \times n}$

$s,e \leftarrow \psi^n$

$t := \text{Round}_\alpha(As+e)$

pk: (A,t)

sk: s

Encrypt(μ):

$r',e' \leftarrow \psi^n$

$f \leftarrow \psi$

$u' := \text{Round}_\alpha(r'A+e')$

$v := \text{Round}_k(r't+f+[q/2]\mu)$

ciphertext: (u',v)

Decrypt(u',v):

$w := v - u's$

$$\mu := \text{Round}_1\left(\frac{w}{[q/2]}\right)$$

Introduces more noise – makes lattice reduction harder

But this noise is deterministic – we choose not to rely on it for hardness

Kyber CCA-KEM Stats

Ring $R_q[X]/(X^{256}+1)$, $q = 2^{13}-2^{9}+1$

	medium	recommended	very high
dimension of A	2 x 2	3 x 3	4 x 4
pk size	736 bytes	1088 bytes	1440 bytes
ciphertext size	832 bytes	1184 bytes	1536 bytes
quantum security	102	161	218
key gen cycles		85K	
enc cycles		125K	
dec cycles		135K	

CRYSTALS: DILITHIUM

DIGITAL SIGNATURE SCHEME

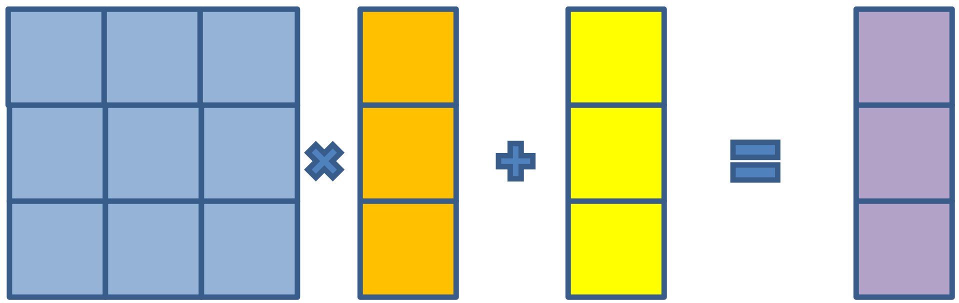
Design Philosophy

- Make it simple to securely implement everywhere – only uniform sampling

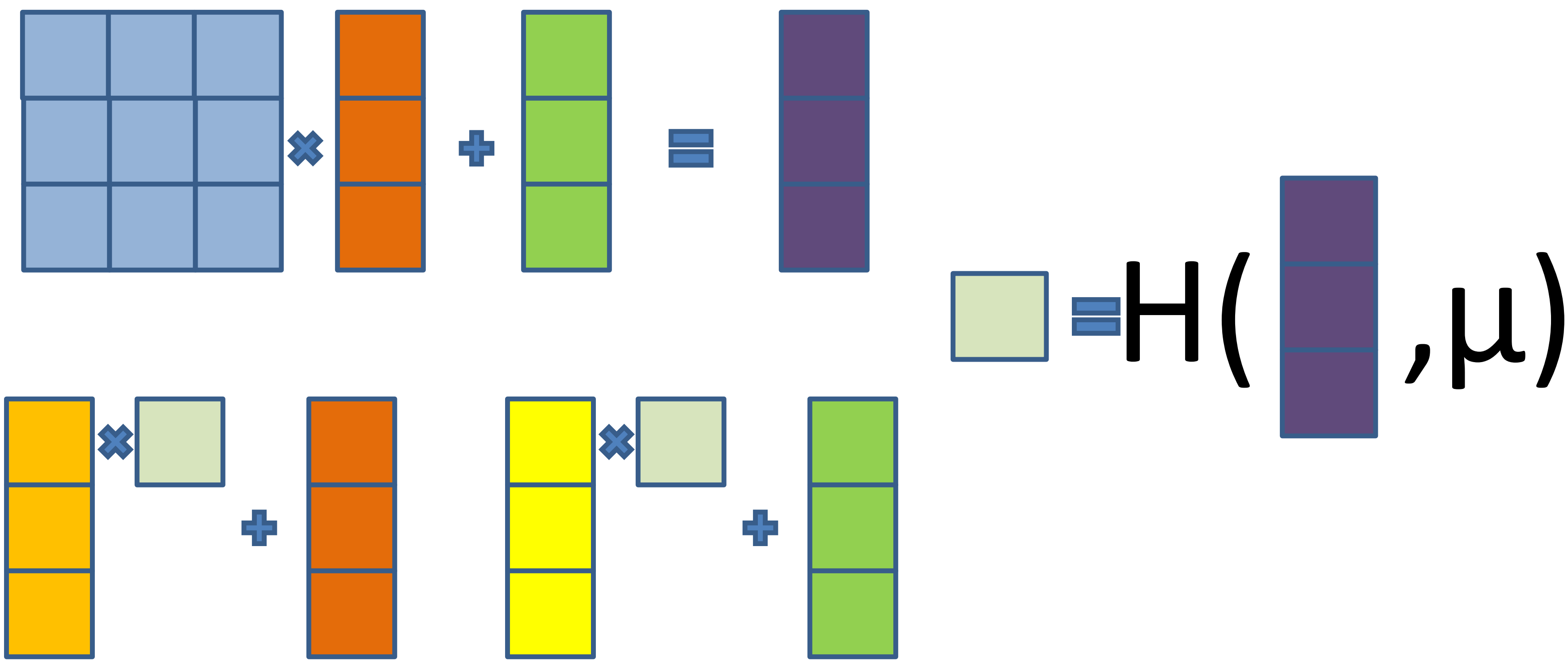
- Public key size is also important – want to minimize (sig size + pk size)

- Make adjusting security levels simple – always operate over the ring $Z_q[X]/(X^{256}+1)$

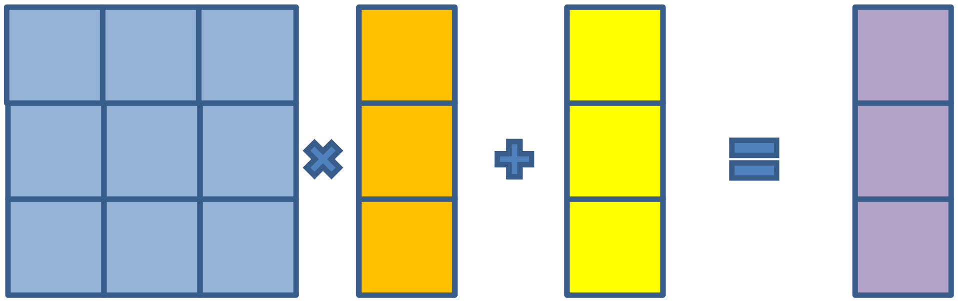
Fiat-Shamir with Aborts [Lyu ‘09]



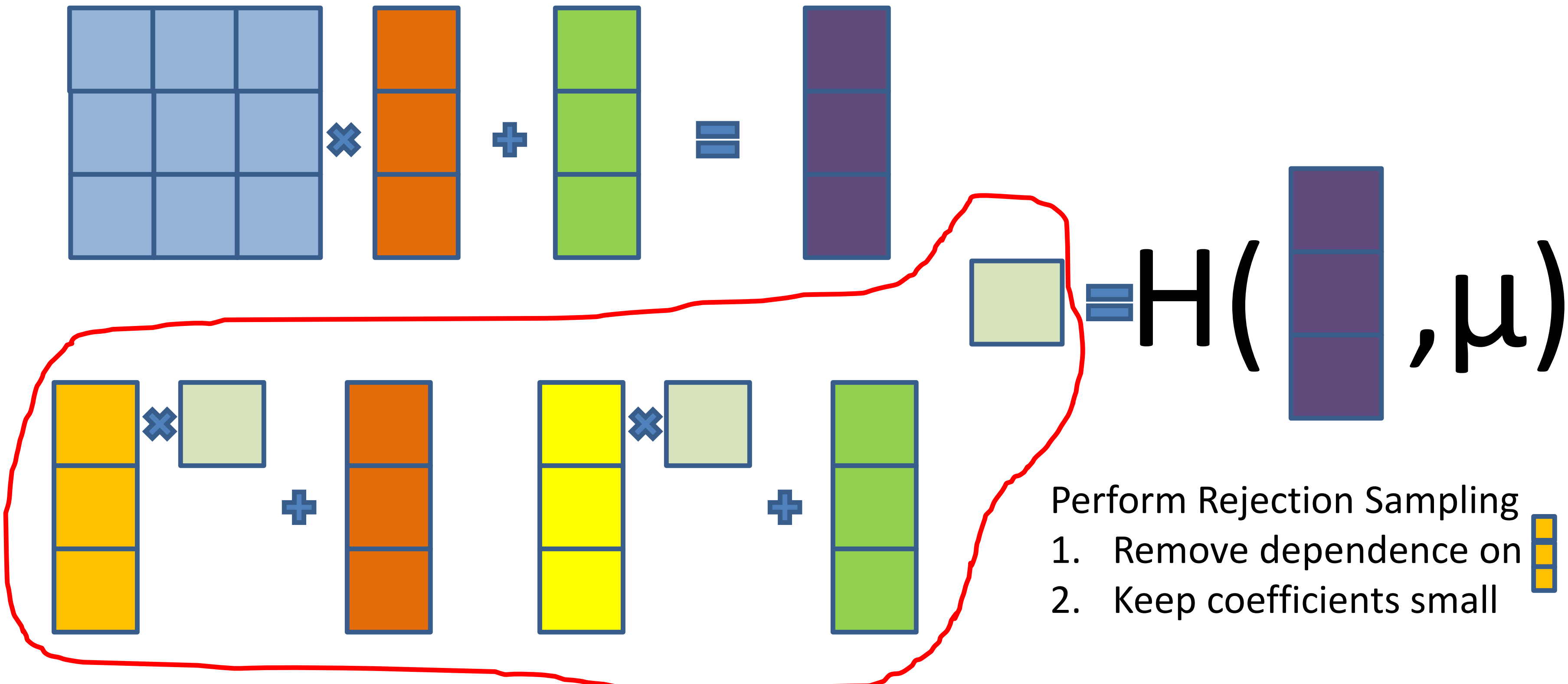
Public Key / Secret Key Generation



Fiat-Shamir with Aborts [Lyu '09]



Public Key / Secret Key Generation



Perform Rejection Sampling

1. Remove dependence on
2. Keep coefficients small

Fiat-Shamir with Aborts [Lyu '09]

$$\mathbf{A}s_1+s_2=\mathbf{t}$$

Sign(μ)

$y_1, y_2 \leftarrow$ D with small coefficients

$c := H(\mathbf{A}y_1+y_2, \mu)$

$z_1 := y_1 + cs_1$, $z_2 := y_2 + cs_2$

RejectionSample(z_1, z_2, cs_1, cs_2)

Signature = (z_1, z_2, c)

Verify(z_1, z_2, c, μ)

Check that z_1, z_2 have small coefficients

and

$c=H(\mathbf{A}z_1+z_2 - c\mathbf{t}, \mu)$

Security Proof

Can simulate signing (by programming H) because the distribution (z_1, z_2, c) is independent of the secret key.

Can extract two signatures such that

$$\mathbf{A}z_1 + z_2 - c\mathbf{t} = \mathbf{A}z_1' + z_2' - c'\mathbf{t}$$

$$\mathbf{A}(z_1 - z_1') + (z_2 - z_2') - (c - c')\mathbf{t} = \mathbf{0}$$

Found a short vector in a lattice

Observations

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') + (\mathbf{z}_2 - \mathbf{z}_2') - (c - c')\mathbf{t} = \mathbf{0}$$

$$\Downarrow$$

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') - (c - c')\mathbf{t} \approx \mathbf{0}$$

Still found a short vector… but now don't have to output $\mathbf{z}_2$ → signature shrunk by about 50% [GLP '12, BG '14]

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') - (c - c')\mathbf{t} \approx \mathbf{0}$$

$$\Downarrow$$

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') - (c - c')(\mathbf{t}_1 + \mathbf{t}_0) \approx \mathbf{0}$$

High-Order Bits of **t**

$$\Downarrow$$

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_1') - (c - c')\mathbf{t}_1 \approx \mathbf{0}$$

Still found a short vector… but now don't have to have $\mathbf{t}_0$ in the public key → public key shrunk by > 50% [DLLSSS '17]

Dilithium Sketch

$\mathbf{A}$:=XOF(ρ), $\mathbf{t}$:=$\mathbf{A}s_1+s_2$

Public key: ρ,$\mathbf{t}_1$

Sign(μ)

y ← D with uniform small coefficients

c := H(HighBits($\mathbf{A}$y), μ)

z := y + cs_1

RejectionSample(z, cs_1, cs_2)

(Must hold: HighBits($\mathbf{A}$y)=HighBits($\mathbf{A}$z-c$\mathbf{t}$))

Create a hint **h** such that

HighBits($\mathbf{A}$z-c$\mathbf{t}_1$) & **h** → HighBits($\mathbf{A}$z-c$\mathbf{t}$)

Signature = (z, **h**, c)

Verify((z, **h**, c), μ)

Use $\mathbf{A}$z-c$\mathbf{t}_1$ and **h** to get

w:= HighBits($\mathbf{A}$z-c$\mathbf{t}$)

Check that z has small coefficients

and

c=H(**w**,μ)

Dilithium Sketch

$\mathbf{A}$:=XOF(ρ), $\mathbf{t}:=\mathbf{A}\mathbf{s}_1+\mathbf{s}_2$

Public key: ρ,$\mathbf{t}_1$

Sign(μ)

$\mathbf{y}$ ← D with uniform small coefficients

c := H(HighBits($\mathbf{A}\mathbf{y}$), μ)

$\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$

RejectionSample($\mathbf{z}$, $c\mathbf{s}_1$, $c\mathbf{s}_2$)

(Must hold: HighBits($\mathbf{A}\mathbf{y}$)=HighBits($\mathbf{A}\mathbf{z}$-c$\mathbf{t}$))

Create a hint **h** such that

HighBits($\mathbf{A}\mathbf{z}$-c$\mathbf{t}_1$) & **h** → HighBits($\mathbf{A}\mathbf{z}$-c$\mathbf{t}$)

Signature = ($\mathbf{z}$, **h**, c)

100 bytes allows to save over 2000 bytes in the pk

Verify(($\mathbf{z}$, **h**, c), μ)

Use $\mathbf{A}\mathbf{z}$-c$\mathbf{t}_1$ and **h** to get

w:= HighBits($\mathbf{A}\mathbf{z}$-c$\mathbf{t}$)

Check that $\mathbf{z}$ has small coefficients

and

c=H(**w**,μ)

Dilithium Stats

Ring $R_q[X]/(X^{256}+1)$, $q = 2^{23} - 2^{13} + 1$

	Medium	Recommended	Very High
dimension of A	4 x 3	5 x 4	6 x 5
pk size	1184 bytes	1472 bytes	1760 bytes
sig size	2043 bytes	2700 bytes	3365 bytes
BKZ block size	340	475	595
classical security	100	140	174
quantum security	91	125	158
key gen cycles	160K	250K	320K
signature cycles	640K	1000K	840K
verification cycles	205K	300K	400K

Comparing to BLISS [DDLL '13]

	BLISS	Medium	Recommended
dimension of A		4 x 3	5 x 4
pk size	875 bytes	1184 bytes	1472 bytes
sig size	820 bytes	2043 bytes	2700 bytes
BKZ block size	280	340	475
classical security	claimed 192, why?	100	140
quantum security		91	125

Most practical attack using BKZ 2.0 [CN '11] takes > 2^{192} time

This was a useful number for comparing with current schemes, e.g. RSA, EC-DSA

Now, we want to be more conservative – (e.g. assume exponential-space sieving is OK)

Higher Security BLISS
(back-of-envelope calculations)

- Using $Z[X]/(X^{1024}+1)$ instead of $Z[X]/(X^{512}+1)$
  - Public Key ≈ 2100 bytes
  - Signature ≈ 1700 bytes
  - Security > 160 quantum
- Using $Z[X]/(f(x))$ for with $\deg(f) \approx 768$
  - Public Key ≈ 1500 bytes
  - Signature ≈ 1300 bytes
  - Security ≈ 128 quantum

BLISS vs. Dilithium

=	Public keys around the same size
+ BLISS	Signatures half the size (save≈1.5KB)
+ Dilithium	No Gaussian (rejection) sampling
+ Dilithium	Security easily adjusted (same ring)
+ Dilithium	Based on Module-LWE vs. NTRU
+ Dilithium	Same framework as ZK proofs

Random Oracle Model vs. Quantum Random Oracle Model

H is a cryptographic hash function

Theorem statements of the form:

“If an adversary, having restricted access to H, can break a primitive S then the reduction can either solve some hard problem P or break H.”

H should be chosen such that it can’t be broken by a quantum algorithm.

Black Box Access to H

- Random Oracle Model – give x, receive H(x)
- Quantum Random Oracle – give superposition of $(x_1,\ldots,x_k)$, receive H(superposition$(x_1,\ldots,x_k)$)

Main open question: Is there a “natural” scheme that is ROM-secure, but is QROM insecure?

ROM vs. QROM

- Similar to the ROM vs. Standard model debate

For encryption – getting QROM is cheap

- add 256 bits
- increases ciphertext by 3%

For signatures – getting QROM is more expensive

- use “Katz-Wang” idea [AFLT ‘12], [TESLA] over rings
- increases signature size by a factor of 2, public key by a factor of 15, and around 10 times slower
- signature + pk size approaches hash-based signatures

Looking Ahead

- For more “advanced cryptography” (e.g. privacy applications, e-voting, etc.), we need zero-knowledge proofs
- Prove knowledge of short s_1,s_2 such that $As_1+s_2=t$

- Same “Fiat-Shamir with Aborts” technique
- Bimodal Gaussians from BLISS don’t help much (in BLISS, A is picked such that $As_1+s_2=0$)

CONCLUSIONS

If You Want Quantum Security Now

For encryption / key exchange:

- Use Kyber
- Very, very good chance that it's fine
- If some parameters need adjusting later, it's very easy

For digital signatures

- Not crucial at this point for many applications
- If you're signing something for the long-term future, and 40KB sigs is not a problem, use (stateless) hash-based sigs e.g. SPHINCS
- If you need something smaller, could use Dilithium

Research Directions

- Cryptanalysis!!!
- Understand whether QROM is relevant in practical attacks and threatens Fiat-Shamir
  - If yes, then:
    - We could consider hash-and-sign signatures. They're small, but a lot of Gaussian sampling and floating-point arithmetic
    - Or just do hash-based signatures and that's it
    - Zero-knowledge proofs will be quite impractical
  - If things remain as they are, then:
    - Create practical advanced primitives – lots of work to do here!