Homomorphic Encryption
Tutorial

Shai Halevi — IBM
CRYPTO 2011
Wouldn’t it be nice to be able to...

- Encrypt my data before sending to the cloud
- While still allowing the cloud to search/sort/edit/... this data on my behalf
- Keeping the data in the cloud in encrypted form
  - Without needing to ship it back and forth to be decrypted
Computing on Encrypted Data

Wouldn’t it be nice to be able to...

- Encrypt my queries to the cloud
- While still allowing the cloud to process them
- Cloud returns encrypted answers
  - that I can decrypt
Computing on Encrypted Data

$skj#hS28ksytA@ ...
Computing on Encrypted Data

$\text{kjh9*mslt@na0} \text{ &maXxjq02bflx}\ m^00a2nm5,A4.\ pE.abxp3m58bsa\ (3saM%w,snanbanq~mD=3akm2,AZ,ltnhde83|3mz{ndewiunb4}gnbTa*} \text{kjew^bw|mdns0}$
Organization of the Tutorial

Two parts with a 10-minute break in between

First part quite high-level
  - Lots of pictures/animations on the slides
  - Covers the [Gentry 2009] blueprint

Second part more algebraic
  - Lots of formulas on the slides
  - Covers newer constructions [GH’11, BV’11, BGV’11] (April 2011 and on)
Some Notations

An encryption scheme: (KeyGen, Enc, Dec)

Plaintext-space = \{0,1\}

\((pk,sk) \leftarrow \text{KeyGen}(\$), \ c \leftarrow \text{Enc}_{pk}(b), \ b \leftarrow \text{Dec}_{sk}(c)\)

Semantic security [GM’84]:

\((pk, \text{Enc}_{pk}(0)) \approx (pk, \text{Enc}_{pk}(1))\)

\(\approx\) means indistinguishable by efficient algorithms
Homomorphic Encryption (FHE)

\[ H = \{ \text{KeyGen, Enc, Dec, Eval} \} \]

\[ c^* \leftarrow \text{Eval}_{pk}(f, c) \]

**Homomorphic:** \( \text{Dec}_{sk}(\text{Eval}_{pk}(f, \text{Enc}_{pk}(x))) = f(x) \)

\( c^* \) may not look like a “fresh” ciphertext

As long as it decrypts to \( f(x) \)

**Compact:** Decrypting \( c^* \) easier than computing \( f \)

Otherwise we could use \( \text{Eval}_{pk}(f, c) = (f, c) \) and \( \text{Dec}_{sk}(f, c) = f(\text{Dec}_{sk}(c)) \)

\[ |c^*| \text{ independent of the complexity of } f \]
Some examples:

"Raw RSA": $c \leftarrow x^e \mod N$ ($x \leftarrow c^d \mod N$)

$x_1^e \times x_2^e = (x_1 \times x_2)^e \mod N$

GM84: $\text{Enc}(0) \in_R \text{QR}$, $\text{Enc}(1) \in_R \text{QNR}$ (in $\mathbb{Z}_N^*$)

$\text{Enc}(b_1) \times \text{Enc}(b_2) = \text{Enc}(b_1 \oplus b_2) \mod N$
More Privacy Homomorphisms

- Mult-mod-p [ElGamal’84]
- Add-mod-N [Pallier’98]
- Quadratic-polys mod p [BGN’06]
- Branching programs [IP’07]
- A different type of solution for any circuit [Yao’82,...]
  - Not compact, ciphertext grows with circuit complexity
  - Also NC1 circuits [SYY’00]
(x,+)-Homomorphic Encryption

It will be really nice to have...

- Plaintext space $\mathbb{Z}_2$ (w/ ops +,x)
- Ciphertexts live in algebraic ring $R$ (w/ ops +,x)
- Homomorphic for both + and x
  - $\text{Enc}(b_1) + \text{Enc}(b_2)$ in $R = \text{Enc}(b_1 + b_2 \mod 2)$
  - $\text{Enc}(b_1) \times \text{Enc}(b_2)$ in $R = \text{Enc}(b_1 \times b_2 \mod 2)$

→ Can compute any function on the encryptions
  - Since every binary function is a polynomial
  - Won’t get exactly this, but it’s a good motivation
The [Gentry 2009] Blueprint
The [Gentry 2009] blueprint

Evaluate any function in four “easy” steps

- **Step 1: Encryption from linear ECCs**
  - Additive homomorphism

- **Step 2: ECC lives inside a ring**
  - Also multiplicative homomorphism
  - But only for a few operations (low-degree poly’s)

- **Step 3: Bootstrapping**
  - Few ops (but not too few) ➞ any number of ops

- **Step 4: Everything else**
  - “Squashing” and other fun activities

June 16, 2011

Error-Correcting Codes (not Elliptic-Curve Cryptography)
Step 1: Encryption from Linear ECCs

For “random looking” codes, hard to distinguish close/far from code

Many cryptosystems built on this hardness
E.g., [McEliece’78, AD’97, GGH’97, R’03,...]
Step 1: Encryption from Linear ECCs

- **KeyGen:** choose a “random” Code
  - Secret key: “good representation” of Code
    - Allows correction of “large” errors
  - Public key: “bad representation” of Code
    - Can generate “random code-words”
    - Hard to distinguish close/far from the code

- \( \text{Enc}(0) \): a word close to Code
- \( \text{Enc}(1) \): a random word
  - Far from Code (with high probability)
Example: Integers mod $p$ [vDGHV 2010]

- Code determined by a secret integer $p$
  - Codewords: multiples of $p$
- Good representation: $p$ itself
- Bad representation:
  - $N = pq$, and also many $x_i = pq_i + r_i$
- Enc(0): subset-sum($x_i$’s)+$r$ mod $N$
  - $r$ is new noise, chosen by encryptor
- Enc(1): random integer mod $N$
Both Enc(0), Enc(1) close to the code
- Enc(0): distance to code is even
- Enc(1): distance to code is odd
- Security unaffected when p is odd

In our example of integers mod $p$:
- $Enc(b) = 2(r+\text{subset-sum}(x_i's)) + b \mod N$
  
  
  $= \kappa p + 2(r+\text{subset-sum}(r_i's)) + b$

- $Dec(c) = (c \mod p) \mod 2$
Additive Homomorphism

\[ c_1 + c_2 = (\text{codeword}_1 + \text{codeword}_2) + (2r_1 + b_1) + (2r_2 + b_2) \]

\[ \text{codeword}_1 + \text{codeword}_2 \in \text{Code} \]

\[ (2r_1 + b_1) + (2r_2 + b_2) = 2(r_1 + r_2) + b_1 + b_2 \]

is still small

If \[ 2(r_1 + r_2) + b_1 + b_2 < \text{min-dist}/2 \], then

\[ \text{dist}(c_1 + c_2, \text{Code}) = 2(r_1 + r_2) + b_1 + b_2 \]

\[ \Rightarrow \text{dist}(c_1 + c_2, \text{Code}) \equiv b_1 + b_2 \pmod{2} \]

Additively-homomorphic while close to \text{Code}
Step 2: Code Lives in a Ring

What happens when multiplying in \textbf{Ring}:

\[ c_1 \cdot c_2 = (\text{codeword}_1 + 2r_1 + b_1) \cdot (\text{codeword}_2 + 2r_2 + b_2) \]

\[ = \text{codeword}_1 \cdot X + Y \cdot \text{codeword}_2 + (2r_1 + b_1) \cdot (2r_2 + b_2) \]

If:

\[ \text{codeword}_1 \cdot X + Y \cdot \text{codeword}_2 \in \text{Code} \]

\[ (2r_1 + b_1) \cdot (2r_2 + b_2) < \text{min-dist/2} \]

Then

\[ \text{dist}(c_1 c_2, \text{Code}) = (2r_1 + b_1) \cdot (2r_2 + b_2) = b_1 \cdot b_2 \mod 2 \]
Secret-key is \(p\), public-key is \(N\) and the \(x_i\)'s

\[ c_i = \text{Enc}_{pk}(b_i) = 2(r + \text{subset-sum}(x_i's)) + b \mod N \]

\[ = k_ip + 2r_i+b_i \]

\[ \text{Dec}_{sk}(c_i) = (c_i \mod p) \mod 2 \]

\[ c_1 + c_2 \mod N = (k_1p + 2r_1 + b_1) + (k_2p + 2r_2 + b_2) - kqp \]

\[ = k'p + 2(r_1 + r_2) + (b_1 + b_2) \]

\[ c_1 c_2 \mod N = (k_1p + 2r_1 + b_1)(k_2p + 2r_2 + b_2) - kqp \]

\[ = k'p + 2(2r_1 r_2 + r_1 b_2 + r_2 b_1) + b_1 b_2 \]

Additive, multiplicative homomorphism

As long as noise < \(p/2\)
We need a linear error-correcting code $\mathcal{C}$

- With “good” and “bad” representations
- $\mathcal{C}$ lives inside an algebraic ring $\mathbb{R}$
- $\mathcal{C}$ is an ideal in $\mathbb{R}$
- Sum, product of small elements in $\mathbb{R}$ is still small

Can find such codes in Euclidean space

- Often associated with lattices

Then we get a “somewhat homomorphic” encryption, supporting low-degree polynomials

- Homomorphism while close to the code
[G 2009] Polynomial Rings
- Security based on hardness of “Bounded-Distance Decoding” in ideal lattices

[vDGHV 2010] Integer Ring
- Security based on hardness of the “approximate-GCD” problem

[GHV 2010] Matrix Rings (only partial solution)
- Only quadratic polynomials, security based on hardness of “Learning with Errors” (LWE)

[BV 2011a] Polynomial Rings
- Security based on “ring LWE”
So far, can evaluate low-degree polynomials

\[ P(x_1, x_2, \ldots, x_t) \]
So far, can evaluate low-degree polynomials

\[ y = P(x_1, x_2, \ldots, x_n) \]

when \( x_i \)'s are “fresh”

But \( y \) is an “evaluated ciphertext”

Can still be decrypted

But eval \( Q(y) \) will increase noise too much

“Somewhat Homomorphic” encryption (SWHE)
Step 3: Bootstrapping

So far, can evaluate low-degree polynomials

$P(x_1, x_2, ..., x_t)$

Bootstrapping to handle higher degrees

We have a noisy evaluated ciphertext $y$

Want to get another $y$ with less noise
For ciphertext \( c \), consider \( D_c(sk) = \text{Dec}_{sk}(c) \)

Hope: \( D_c(*) \) is a low-degree polynomial in \( sk \)

Include in the public key also \( \text{Enc}_{pk}(sk) \)

Homomorphic computation applied only to the “fresh” encryption of \( sk \)
Step 3: Bootstrapping

Similarly define $M_{c_1, c_2}(sk) = \text{Dec}_{sk}(c_1) \cdot \text{Dec}_{sk}(c_2)$

Homomorphic computation applied only to the “fresh” encryption of $sk$
Cryptosystems from [G’09, vDGHV’10, BG’11a] cannot handle their own decryption.

Tricks to “squash” the decryption procedure, making it low-degree:

- Nontrivial, requires putting more information about the secret key in the public key.
- Requires yet another assumption, namely hardness of the Sparse-Subset-Sum Problem (SSSP).
- I will not talk about squashing here.
SWHE schemes may be reasonable
But bootstrapping is inherently inefficient
  Homomorphic decryption for each multiplication
  Asymptotically, overhead of at least $\tilde{O}(\lambda^{3.5})$
Best implementation so far is [GH 2011a]
  Implemented a variant of [Gentry 2009]
  Public key size $\sim$ 2GB
  Bootstrapping takes 3-30 minutes
Similar performance also in [CMNT 2011]
  Implemented the scheme from [vDGHV’10]
Beyond the Blueprint
Bootstrapping without squashing
Hybrid of SWHE and MHE schemes
  MHE = Multiplicative HE (e.g., Elgamal)
Express decryption as a “restricted depth-3” $\Sigma\Pi\Sigma$ arithmetic circuit
Switch to MHE for the middle $\Pi$ level
  All necessary MHE ciphertexts found in public key
Translate back to SWHE for the top $\Sigma$ level
  SWHE evaluates MHE decryption, not own decryption
No need for squashing, SSSP
FHE without squashing, security based on Learning-with-Errors (LWE), or ring-LWE

Main innovation: multiplicative homomorphism without a ring structure

A host of new techniques/tricks, can be used for further improvements
Learning with Errors (LWE) [Regev 2005]

Hard to solve linear equations with noise

Given:

- \( b \) is a random vector in \( \mathbb{Z}_q^m \), or
- \( b \) is close to the row-space of \( A \) (distance < \( \beta q \))

\[ b = s A + e \text{ for random } s \in \mathbb{Z}_q^n \text{ and random short } e \in \mathbb{Z}_q^m \]

Parameters: \( n \) (dim), \( q \geq \text{poly}(n) \) (modulus), \( \beta \leq 1/\text{poly}(n) \) (noise magnitude), \( m = \text{poly}(n) \)

[Regev’05, Peikert’09]: As hard as some worst-case lattice problems in dim \( n \) (for certain range of params)
The [BV’11b] Construction

- Bit-by-bit encryption, plaintext is a bit $b$
- Think of it as symmetric encryption for now
- Secret-key $s$, ciphertext $c$, are vectors in $\mathbb{Z}_q^n$
  - Simplifying convention: $s_1 = 1$, i.e., $s = (1\mid t)$
- Decryption is $b = (\langle s, c \rangle \mod q) \mod 2$
  - $\langle s, c \rangle \mod q$ is small, absolute value $\leq \beta q$
- In other words:
  - Ciphertexts are “close” to space orthogonal to $s$
  - Plaintext encoded in parity of “distance”
  - “distance” is the size of $\langle s, c \rangle \mod q$
This is an instance of encryption from linear ECCs, additive homomorphism is “for free”

As long as things remain close to the code

But how to multiply?

Ciphertexts are vectors, not ring elements

Tensor product (??)

\[ \mathbf{M} = \mathbf{u} \otimes \mathbf{v}, \quad M_{ij} = u_i \cdot v_j \mod q \]

Can decrypt \( \mathbf{M}(!) \),

\[ s (\mathbf{u} \otimes \mathbf{v}) s^t = <s, \mathbf{u}> \cdot <s, \mathbf{v}> \mod q \]

If no wraparound then

\[ (s (\mathbf{u} \otimes \mathbf{v}) s^t \mod q) = (<s, \mathbf{u}> \mod q) \cdot (<s, \mathbf{v}> \mod q) \]

So \( (s (\mathbf{u} \otimes \mathbf{v}) s^t \mod q) \mod 2 = \text{Dec}_s(\mathbf{u}) \cdot \text{Dec}_s(\mathbf{v}) \)
Multiplying More than Once?

$s(u \otimes v)s^t$ is a bilinear form in $s$, so linear in $s \otimes s$

Opening $u \otimes v$, $s \otimes s$ into vectors, we get

$s(u \otimes v)s^t = \langle \text{vec}(s \otimes s), \text{vec}(u \otimes v) \rangle$

Denote $s^* = \text{vec}(s \otimes s)$, $c^* = \text{vec}(u \otimes v)$, then:

Dec$_{s^*}(c^*) = (\langle s^*, c^* \rangle \mod q) \mod 2$

$\langle s^*, c^* \rangle \mod q$ is still quite small, $\leq (\beta q)^2 << q$

We can repeat the process

But dimension is squared, $n \rightarrow n^2 \rightarrow n^4 \rightarrow n^8 \ldots$

so can repeat only a constant number of times
Reducing the Dimension

We have an “extended ciphertext” $c^*$ with respect to “extended secret key” $s^* = \text{vec}(s \otimes s)$.

Want a low-dimension ciphertext $c'$ with respect to a “standard secret key” $s'$.

Maybe $s' = s$, maybe not.

Key idea: publish “an encryption” of $s^*$ under $s'$ to enable the translation.

Hopefully just a matrix $M(s^* \rightarrow s') \in \mathbb{Z}_q^\text{dim}(s') \times \text{dim}(s^*)$, so that $c' = M \cdot c^* \in \mathbb{Z}_q^\text{dim}(s')$. 


Recall $s'=(1|t')$, so $s'M = t'A + b = 2e + s^*$

Let $c' = M \cdot c^* \in \mathbb{Z}_q^{\dim(t') \times \dim(s^*)}$, then mod $q$ we have:

$$<s', c'> \equiv s'Mc^* \equiv 2e + s^*, c^* > \equiv <s^*, c^*> + 2<e, c^*>$$

If only $c^*$ was short, then $2<e, c^*>$ was small, so

$$(<2e + s^*, c^*> \mod q) = (<s^*, c^*> \mod q) + 2<e, c^*>$$

Hence $$(<s', c'> \mod q) \equiv (<s^*, c^*> \mod q) \mod 2$$

So $\text{Dec}_{s'}(c') = \text{Dec}_{s^*}(c^*)$
Can we Make $c^*$ Short?

Want to “represent” the long vector $c^*$ by some short vector $c'$, perhaps in higher dimension

Example: $c^*=(76329, 31692, 43870)$

- $l_2$-norm $\sim 90000$

represented by $c'=(7,6,3,2,9, 3,1,6,9,2, 4,3,8,7,0)$

- $l_2$-norm only $\sim 21$

Later we will use binary rather than decimal

Note that we have a “linear relation”:

$$c^* = 10^4 \cdot c'_{1,6,11} + \cdots + 10 \cdot c'_{4,9,14} + c'_{5,10,15}$$
Can we Make $c^*$ Short?

Denote $c^*=(c^*_1, \ldots, c^*_k)$, i.e., $c^*_i$ is the $i$'th entry

Let $c^*_{i l} \ldots c^*_{i 0}$ be binary representation of $c^*_i$

$$c^*_i = \sum_{j=0}^{l} 2^j c^*_{i j}$$

Let $b_j$ be the vector of $j$'th bits $b_j=(c^*_{1 j}, \ldots, c^*_{k j})$

so $c^* = \sum_{j=0}^{l} 2^j b_j$, and $\langle s^*, c^* \rangle = \sum_{j=0}^{l} 2^j \langle s^*, b_j \rangle$

Let $s^{**}=\text{PowersOf2}_q(s^*)= (s^*|2s^*|4s^*|\ldots|2^l s^*) \mod q$

and $c^{**}=\text{BitDecomp}(c^*) = (b_0|b_1 \ | b_2 \ | \ldots \ | b_l)$

Then $\langle s^{**}, c^{**} \rangle \equiv \langle s^*, c^* \rangle \pmod{q}$

$c^{**}$ is short (in $l_2$-norm), it is a 0-1 vector
Dimension Reduction (Key-Switching)

Publish the matrix $M(s^{**} \rightarrow s') \in \mathbb{Z}_q^{\dim(s') \times \dim(s^{**})}$

Given the expanded ciphertext $c^*$

- Compute the “doubly expanded” $c^{**}$
- Set $c' = M \cdot c^{**} \mod q$

We know that $\langle s^{**}, c^{**} \rangle \equiv \langle s^*, c^* \rangle \pmod{q}$

Also $\langle s', c' \rangle \equiv \langle s^{**}, c^{**} \rangle + 2\langle e, c^{**} \rangle \pmod{q}$

$(\langle s^*, c^* \rangle \mod q)$ is small and so is $2\langle e, c^{**} \rangle$ hence

$(\langle s', c' \rangle \mod q) = (\langle s^*, c^* \rangle + 2\langle e, c^{**} \rangle \mod q)$

Last equality is over the integers

$\Rightarrow \text{Dec}_{s'}(c') = \text{Dec}_{s^*}(c^*)$
Under LWE, cannot tell $M(s^* \rightarrow s')$ from random

Even if you know $s^*$ (but not $s'$)

Assuming $q$ is odd

$Pf$: if $(A, r) \approx (A, tA+e)$ then $(2A, 2r) \approx (2A, 2tA+2e)$

For odd $q$:

$(2A, 2r) \equiv (A, r)$,

$(2A, 2tA+2e) \equiv (A, tA+2e)$

$\equiv$ means that these distributions are identical

We get $(A, r) \approx (A, tA+2e)$

It follows that $(A, r) \equiv (A, r+s^*) \approx (A, tA+2e+s^*)$
The [BV’11b] “Leveled SWHE”
(Key-size $\geq$ linear in depth of circuits to evaluate)

**KeyGen**: choose random $s_0, s_1, \ldots, s_d \in \mathbb{Z}_q^n$
- First entry in each $s_i$ is 1
- Public key has matrices $M_0 = M(0 \rightarrow s_0)$ and $M_{i+1} = M(s_i \rightarrow s_{i+1})$ for $i=0,1,\ldots,d-1$
- Then $s_0M_0 = 2e_0$, and $s_iM_i = 2e_i + s_{i-1}^*$

**Enc(b)**: $r \in \mathbb{R}\{0,1\}^m$, $c \leftarrow M_0r + [b,0,\ldots,0]$, output $(c,0)$

**Dec(c,i)**: Recover $b \leftarrow \langle s_i,c \rangle \mod q \mod 2$
- For level-0: $\langle s_0,c \rangle = s_0M_0r + b = 2\langle e_0,r \rangle + b$
- $e_0,r$ are short so $2\langle e_0,r \rangle \ll q$, hence no wraparound
The [BV’11b] “Leveled SWHE”

Ciphertexts in same level can be added directly

To multiply two level-$i$ ciphertexts $(c_1,i),(c_2,i)$

Compute the extended $c^* = \text{vec}(c_1 \otimes c_2)$, the “doubly extended” $c^{**}$, and set $c’ \leftarrow M_i c^{**}$

$(c’,i+1)$ is a level-$(i+1)$ ciphertext

Semantic-security follows because:

Under LWE, the $M_i$’s are pseudo-random

If they were random then ciphertexts would have no information about the encrypted plaintexts

By leftover hash lemma
The “noise” in a ciphertext \((c, i)\) is \(\langle s_i, c \rangle \mod q\)

- Noise magnitude roughly doubles on addition, get squared on multiplication
- Can only evaluate log-depth circuits before the noise magnitude exceeds \(q\)

How to evaluate deeper circuits?
- Squash & bootstrap,
- Chimeric & bootstrap,
- or an altogether new technique...
Modulus Switching

Converting $c, s$ into $c', s'$ s.t. for some $p < q$
$(<s', c'> \mod p) \equiv (<s, c> \mod q) \mod 2$

[BV’11b]: Use with $p \ll q$ to reduce decryption complexity, can bootstrap without squashing
- Modulus-switching & key-switching combined
- The resulting $c'$ can be decrypted, but cannot participate in any more homomorphic operations

[BGV’11] Use with $p < q$ to reduce the noise, can compute deeper circuits w/o bootstrapping
- Roughly just by scaling, $c' \leftarrow \text{round}(p/q \cdot c)$
- Rounding “appropriately”
Let $p < q$ be odd integers, $c,s \in \mathbb{Z}_q^n$ such that

\[ |\langle s, c \rangle \mod q| < q/2 - q/p \cdot \|s\|_1 \]

$\|s\|_1$ is the $l_1$ norm of $s$

Let $c' = \text{rnd}_c(p/q \cdot c)$, where $\text{rnd}_c(\cdot)$ rounds each entry up or down so that $c' \equiv c \pmod{2}$

Then (i) $\langle s, c' \rangle \mod p \equiv \langle s, c \rangle \mod q \pmod{2}$

and (ii) $|\langle s, c' \rangle \mod p| \leq \frac{p}{q} \cdot |\langle s, c \rangle \mod q| + \|s\|_1$
Modulus Switching – Main Lemma

Proof:

For some κ, \( \langle s, c \rangle \mod q = \langle s, c \rangle - \kappa q \in \left[ \frac{-q}{2}, \frac{q}{2} \right] \)

Actually in a smaller interval
\[ \langle s, c \rangle - \kappa q \in \left[ \frac{-q}{2} + \frac{q}{p} \ l_1 s, \frac{q}{2} - \frac{q}{p} \ l_1 s \right] \]

Multiplying by \( \frac{p}{q} \) we get
\[ \frac{p}{q} \langle s, c \rangle - \kappa p \in \left[ \frac{-p}{2} + \frac{p}{2} l_1 s, \frac{p}{2} - \frac{p}{2} l_1 s \right] \]

Replacing \( \frac{p}{q} c \) by \( c' = \text{rnd}_c \left( \frac{p}{q} c \right) \), adds error \( \leq l_1 s \):
\[ \langle s, c' \rangle - \kappa p \in \left[ \frac{-p}{2}, \frac{p}{2} \right], \text{ so } \langle s, c' \rangle - \kappa p = \langle s, c' \rangle \mod p \]

This also proves Part (ii)
Modulus Switching – Main Lemma

Proof:

We know that \( <s,c> \mod q = <s,c> - \kappa q \) and
\( <s,c'> \mod p = <s,c'> - \kappa p \) for the same \( \kappa \)

Since \( p,q \) are odd then \( \kappa p \equiv \kappa q \) (mod 2)

Since \( c' \equiv c \) (mod 2) then \( <s,c'> \equiv <s,c> \) (mod 2)

\( <s,c'> \mod p = <s,c'> - \kappa p \)
\( \equiv <s,c> - \kappa q \) (mod 2)
\( = (<s,c> \mod q) \)

This proves part (i)
Making $s$ Small

- If $s$ is random in $\mathbb{Z}_q^n$ then $\|s\|_1 > q$
- Luckily [ACPS 2009] proved that LWE is hard even when $s$ is a random short vector chosen from the same distribution as the noise $e$
- So we use this distribution for the secret keys
- Alternatively, we could have used the trick with BitDecomp() and PowersOf2()
Modulus Switching

Example: $q=127$, $p=29$, $c=(175,212)$, $s=(2,3)$

$<s,c> \mod q = 986 - 8 \times 127 = -30$

$p/q \cdot c \approx (39.9, 48.4)$

To get $c' \equiv c \pmod{2}$ we round down both entries

$c'=(39,48)$

$<s,c'> \mod p = 222 - 8 \times 29 = -10$

Indeed $\kappa=8$ in both cases, $-10 \equiv -30 \pmod{2}$

The noise magnitude decreased from 30 to 10

But the relative magnitude increased, $\frac{10}{29} > \frac{30}{127}$
How Does Modulus-Switching Help?

- Start with large modulus $q_0$, small noise $\eta \ll q_0$
- After 1st multiplication, noise grows to $\approx \eta^2$
- Switch the modulus to $q_1 \approx q_0 / \eta$,
  - Noise similarly reduced to $\approx \eta^2 / \eta = \eta$
- After next multiplication layer, noise again grows to $\approx \eta^2$, switch to $q_2 \approx q_1 / \eta$ to reduce it back to $\eta$
- Keep switching moduli after each layer
  - Setting $q_{i+1} \approx q_i / \eta$
  - Until last modulus is too small, $q_d / 2 \leq \eta$
Example: \( q_0 \approx \eta^5 \)

<table>
<thead>
<tr>
<th>Fresh ciphertexts</th>
<th>Using mod-switching</th>
<th>Without mod-switching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise</td>
<td>Modulus</td>
<td></td>
</tr>
<tr>
<td>( \eta )</td>
<td>( \eta^5 )</td>
<td>( \eta )</td>
</tr>
<tr>
<td>Level-1, degree=2</td>
<td>( \eta )</td>
<td>( \eta^4 )</td>
</tr>
<tr>
<td>Level-2, degree=4</td>
<td>( \eta )</td>
<td>( \eta^3 )</td>
</tr>
<tr>
<td>Level-3, degree=8</td>
<td>( \eta )</td>
<td>( \eta^2 )</td>
</tr>
<tr>
<td>Level-4, degree=16</td>
<td>( \eta )</td>
<td>( \eta )</td>
</tr>
</tbody>
</table>

Decryption errors
Use tensor-product for multiplication
Then reduce the dimension with $M(s \rightarrow s')$
  First need to use PowersOf2/BitDecomp
Then reduce the noise by switching modulus
  This works if the secret key $s$ is short
Repeat until modulus is too small
**The [BGV’11] “Leveled FHE”**

- $d$-level circuits, initial noise $\eta$
  
  Also $\tau \triangleq \eta \cdot \text{poly}(n)$ is another parameter

Set odd moduli $q_0, \ldots, q_d$ s.t. $q_i \approx \tau^{d-i+1}$

**Key generation:**

Choose short secret $s_i \in \mathbb{Z}_{q_i}^n$, $i=0,\ldots,d$, first entry=1

- Set $s_i^* = \text{vec}(s_i \otimes s_i) \in \mathbb{Z}_{q_i}^{n^2}$, $s_i^{**} = \text{PowersOf2}_{q_i}(s_i^*) \in \mathbb{Z}_{q_i^{t_i}}$

Public key has $M_0 = M(0 \rightarrow s_0) \in \mathbb{Z}_{q_0}^{n \times t_0}$

and $M_i = M(s_{i-1}^{**} \rightarrow s_i) \in \mathbb{Z}_{q_i}^{n \times t_{i-1}}$

The “short error vector” in $M_i$ is $e_i \in \mathbb{Z}_{q_i^{t_{i-1}}}$

Then $s_0 M_0 = 2e_0 \mod q_0$ and $s_i M_i = 2e_i + s_{i-1}^{**} \mod q_{i-1}$

$t_0 = 3n \log(q_0)$ and $t_i = n^2 \log(q_i)$
The [G’11] “Leveled FHE”

Enc, Dec, and homomorphic addition are the same as in the leveled SWHE

Level-\(i\) ciphertexts are modulo \(q_i\)

To multiply two level-\(i\) ciphertexts, \(c_1, c_2\):

\[
c^* \leftarrow \text{vec}(c_1 \otimes c_2) \in \mathbb{Z}_{q_i}^n, \quad (<s_i^*, c^*> \mod q_i) \equiv b_1 b_2 \pmod{2}
\]

\[
c^{**} \leftarrow \text{BitDecom}(c^*), \quad (<s_i^{**}, c^{**}> \mod q_i) \equiv b_1 b_2 \pmod{2}
\]

\[
c' \leftarrow M_{i+1} c^{**} \mod q_i \quad (<s_{i+1}, c'> \mod q_i) \equiv b_1 b_2 \pmod{2}
\]

\[
c \leftarrow \text{rnd}_c(q_{i+1}/q_i \cdot c'), \quad (<s_{i+1}, c> \mod q_{i+1}) \equiv b_1 b_2 \pmod{2}
\]

Noise in \(c\) is bounded by \((\eta^2 + \text{stuff})/\tau \leq \eta\)
What We Have So Far

- A leveled-FHE:
  - Public-key size at least linear in circuit depth
  - Can handle circuits of arbitrary polynomial depth

- Security based on LWE
  \[
  \frac{1}{\beta} \approx \frac{\text{modulus}}{\text{noise}} = (\text{poly}(n))^\text{depth}
  \]
  - For “interesting” circuits this is more than \(\text{poly}(n)\)

- Modulus gets smaller as we go up the circuit
  - Lower levels somewhat more expensive
Use bootstrapping to recover large modulus
- Size of largest modulus depends on decryption circuit, not the circuits that we evaluate
- Can be made into “pure” FHE (non-leveled), need to assume circular security

Base security on ring-LWE
- LWE over a ring other than $\mathbf{Z}_q$ (e.g., $\mathbf{R} = \mathbf{Z}_q[x]/f(x)$)
- Can use smaller dimension (e.g., dim=2)

Large plaintext space (not just $\mathbf{Z}_2$)
- Must tweak the modulus-switching technique
Variants and Optimizations

- Batching: pack many bits into each ciphertext
  - E.g., using the Chinese Remainders Theorem
  - An operation (+,x) on ciphertext acts separately on each the packed bits

- Combining these optimizations, can reduce the overhead to $\tilde{O}(\lambda)$
  - Compare to $\tilde{O}(\lambda^{3.5})$ for the original blueprint
Many new ideas are at the table now
  Still figuring out what works and what doesn’t
  Looking at recent history, we can expect more new ideas in the next few months/years

Implementation efforts are underway
  Goal: get usable FHE
  At least for some applications
  My personal guess: almost at hand, perhaps only 2-3 years away

Many open problems remain