Homomorphic Encryption from Ring Learning with Errors

Michael Naehrig

Technische Universiteit Eindhoven
michael@cryptojedi.org

joint work with
Kristin Lauter (MSR Redmond)
Vinod Vaikuntanathan (University of Toronto)

MSR Cambridge, 10 January 2012
Homomorphic encryption

Example 1: RSA public key encryption
- Let $n = p \cdot q$, $p \neq q$ primes, $\varphi(n) = (p - 1)(q - 1)$,
- $pk = (n, e)$, $\gcd(e, \varphi(n)) = 1$,
- $sk = d = e^{-1} \mod \varphi(n)$.
- Encrypt message $m \in \mathbb{Z}_n$:
  \[ c = m^e \mod n. \]
- Decrypt ciphertext $c$:
  \[ m = c^d \mod n. \]
- Multiplicative homomorphism:
  If $c_1 = m_1^e \mod n$, $c_2 = m_2^e \mod n$, then
  \[ c_1 \cdot c_2 = (m_1 \cdot m_2)^e \mod n. \]
Homomorphic encryption

Example 2: ElGamal public key encryption in a group \( G = \langle g \rangle \)

- \( \text{sk} = x \in \mathbb{Z}_{|G|} \),
- \( \text{pk} = h = g^x \).
- Encrypt \( m \in G \): choose \( r \in \mathbb{Z}_{|G|} \) at random and compute
  \[
  (c, d) = (g^r, m \cdot h^r).
  \]
- Decrypt: \( m = d \cdot (c^x)^{-1} \).
- Multiplicative homomorphism:
  If \( (c_1, d_1) = (g^{r_1}, m_1 \cdot h^{r_1}), (c_2, d_2) = (g^{r_2}, m_1 \cdot h^{r_2}) \), then
  \[
  (c_1 \cdot c_2, d_1 \cdot d_2) = (g^{r_1} \cdot g^{r_2}, (m_1 \cdot h^{r_1}) \cdot (m_2 \cdot h^{r_2}))
  = (g^{r_1+r_2}, (m_1 \cdot m_2)h^{r_1+r_2}).
  \]
Homomorphic encryption

- Many crypto systems have homomorphic properties: RSA, ElGamal, Benaloh, Paillier, but only provide additive or multiplicative homomorphism, not both.
- With addition and multiplication, can do arbitrary computations.
- First system that could do both: Boneh-Goh-Nissim 2005 many additions and one multiplication (uses pairings).
- Fully homomorphic encryption allows to do arbitrary computations on encrypted data without knowing the secret key,
- in particular it allows doing an arbitrary number of additions and multiplications.
Application scenario

User

\[\xrightarrow{\text{encrypted data}}\]

\[\xrightarrow{\text{encrypted results}}\]

Server

operates on encrypted data: e.g. search, statistics, ...

Server never sees data in the clear.

But does a fully homomorphic encryption scheme exist? And if so, is it efficient?
Gentry proposed the first fully homomorphic encryption scheme in 2009 based on ideal lattices.

- The basis is a somewhat homomorphic encryption scheme that can evaluate low-degree polynomials on encrypted data.
- Ciphertexts are “noisy” and the noise grows slightly during addition and strongly during multiplication.
- If the SWHE scheme can evaluate its own decryption circuit, then a bootstrapping step can refresh ciphertexts by homomorphically decrypting using an encrypted secret key.
- Only works by “squashing” the decryption circuit.
- So far quite inefficient.
Fully homomorphic encryption

- Recently, many improvements, but still inefficient. Implementation (Gentry, Halevi 2011),
  - toy setting: encrypt a bit in 0.2s, recrypt in 6s, public key: 17MB
  - large setting: encrypt in 3min, recrypt in 31min, public key: 2.3GB
- New variants, mostly following Gentry’s blueprint.
- Recent variants based on the LWE problem or RLWE problem.
- Applications might not need fully homomorphic encryption, somewhat homomorphic could be sufficient.
- This talk: somewhat homomorphic encryption scheme by Brakerski and Vaikuntanathan (Crypto 2011) based on RLWE.
The Learning with Errors (LWE) Problem
(Regev 2005)

Let $n \in \mathbb{N}$, $q \in \mathbb{Z}$, and $\chi$ a probability distribution on $\mathbb{Z}$.

Distinguish the following distributions of pairs $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$:

1. **Uniform distribution**
   - Sample $(a_i, b_i) \in \mathbb{Z}_q^{n+1}$ uniformly at random.

2. **LWE distribution**
   - Draw $s \in \mathbb{Z}_q^n$ uniformly at random.
   - Sample $a_i \in \mathbb{Z}_q$ uniformly at random,
   - Sample $e_i \leftarrow \chi$, $\overline{e}_i \in \mathbb{Z}_q$,
   - Set $b_i = \langle a_i, s \rangle + \overline{e}_i$.

The $b_i$ are noisy inner products of random $a_i$ with a secret $s$. 
The Learning with Errors (LWE) Problem
(Regev 2005)

- Regev gave a quantum reduction of certain approximate SVP to LWE, i.e. if one can solve LWE, then there’s a quantum algorithm to solve certain approximate SVP.
- Peikert (2009) gave a reduction using classical algorithms
- Assumption: $q$ prime, $\chi$ is a discrete Gaussian error distribution
Here: special case.

- Let $n = 2^k$, 
  \[ f(x) = x^n + 1 \]
  (2n-th cyclotomic polynomial).

- Define ring 
  \[ R = \mathbb{Z}[x] / (f) \]
  (ring of integers in 2n-th cyclotomic number field).

- Let $q$ be prime, define 
  \[ R_q = R / qR \cong \mathbb{Z}_q[x] / (f). \]

- Let $\chi$ be an error distribution on $R$. 
Distinguish the following distributions of pairs \((a_i, b_i) \in R^2_q:\)

**Uniform distribution on** \(R^2_q\)
- Sample \((a_i, b_i) \in R^2_q\) uniformly at random.

**RLWE distribution**
- Draw \(s \in R_q\) uniformly at random.
- Sample \(a_i \in R_q\) uniformly at random,
- sample \(e_i \leftarrow \chi, \overline{e}_i \in R_q,\)
- set \(b_i = a_i \cdot s + \overline{e}_i.\)

The \(b_i\) are noisy ring (number field) products of random \(a_i\) with a secret \(s.\)
Toy(!) example parameter setting

Let’s take $k = 3$, i.e. $f = x^8 + 1$, $q = 97$.

- A typical (random) element in $R_q$ looks like this:
  \[ a = 27x^7 - 11x^6 - 33x^5 + 41x^4 - 18x^3 - 5x^2 - 37x - 16. \]

- A small element sampled coefficient-wise from a narrow Gaussian, might look like this:
  \[ e = -2x^6 - 2x^3 + 2x^2 - x + 1. \]

- Addition in $R_q$:
  \[
  a + e = 27x^7 - 13x^6 - 33x^5 + 41x^4 - 20x^3 - 3x^2 - 38x - 15,
  \]
  \[
  a + a = -43x^7 - 22x^6 + 31x^5 - 15x^4 - 36x^3 - 10x^2 + 23x - 32.
  \]

- Multiplication in $R_q$:
  \[
  x \cdot a = 27x^8 - 11x^7 - 33x^6 + 41x^5 - 18x^4 - 5x^3 - 37x^2 - 16x
  \]
  \[
  = -11x^7 - 33x^6 + 41x^5 - 18x^4 - 5x^3 - 37x^2 - 16x - 27.
  \]
The Ring Learning with Errors (RLWE) Problem
(Lyubashevsky, Peikert, Regev 2010)

- Believed to be as hard as general LWE problem, i.e. would be solved with the same algorithms.
- Though there’s a lot more structure!
- Recent results indicate RLWE problem easier than LWE, (Schneider 2011 claims in practice speedup is linear in $n$).
- But much more more efficient.
- Smaller keys, very efficient arithmetic in $R_q$.

Can be used to build a fully homomorphic encryption scheme.
In both LWE and RLWE problems, it is okay to sample $s \leftarrow \chi$ (and not uniformly at random).

- Sample until $(a_0, b_0 = a_0s + e_0)$ with $a_0 \in R_q^*$ (invertible).
- For every additional sample $(a, b = as + e)$ consider

\[
(a', b') = (-a_0^{-1}a, b + a'b_0) \\
= (a', as + e + a'(a_0s + e_0)) \\
= (a', as + e - as + a'e_0) = (a', a'e_0 + e)
\]

- If one can solve RLWE with small secret, then one can solve it with uniform secret.

- It is also okay to use small multiples of the error terms, i.e. samples $(a_i, b_i = a_i \cdot s + te_i)$ are still indistinguishable from random. For example, take $t = 2$. 
Somewhat homomorphic encryption  
(Brakerski, Vaikuntanathan 2011)

**SH.Keygen**

- Sample small $s \leftarrow \chi$. Set secret key $sk = s$.

Sample RLWE instance:

- Sample $a_1 \leftarrow R_q$ unif. rand., small error $e \leftarrow \chi$.

Set public key

- $pk = (a_0 = -(a_1 s + te), a_1)$.

In the example setting: $t = 2$

- $s = -x^7 - x^6 - x^5 + x^4 + x^3 + x^2 + x - 1$
- $e = -2x^6 - 2x^3 + 2x^2 - x + 1$
- $a_1 = 27x^7 - 11x^6 - 33x^5 + 41x^4 - 18x^3 - 5x^2 - 37x - 16$
- $a_0 = 10x^7 - 25x^6 + 46x^5 - 37x^4 + 23x^3 + 27x^2 - 43x + 31$
- $pk = (10x^7 - 25x^6 + 46x^5 - 37x^4 + 23x^3 + 27x^2 - 43x + 31, 27x^7 - 11x^6 - 33x^5 + 41x^4 - 18x^3 - 5x^2 - 37x - 16)$.
Somewhat homomorphic encryption
(Brakerski, Vaikuntanathan 2011)

Message space:

\[ R_t = \mathbb{Z}_t[x]/(x^n + 1), \]

\( t \) rel. prime to \( q \), e.g. \( t = 2 \). Encode messages as elements in \( R_q \) with coefficients mod \( t \).

- Can encode \( n \) bits at once.
- For example encode 01011001 as \( m = x^6 + x^4 + x^3 + 1 \).

**SH.Enc**

Given \( pk = (a_0, a_1) \) and a message \( m \in R_q \),

- sample \( u \leftarrow \chi \), and \( g, h \leftarrow \chi \),

Set ciphertext

- \( ct = (c_0, c_1) := (a_0u + tg + m, a_1u + th) \).
Somewhat homomorphic encryption

Example encryption

- Sample small elements
  
  \[ u = -2x^6 + 3x^5 + 2x^3 - x, \]
  \[ g = -x^6 - x^2 + 2x, \]
  \[ h = -x^7 + x^5 + x^4 + x + 1. \]

- From \( \text{pk} = (a_0, a_1) \) as above and \( m = x^6 + x^4 + x^3 + 1 \) compute
  
  \[ c_0 = a_0 \cdot u + 2 \cdot g + m \]
  \[ = 21x^7 + 2x^6 + 10x^5 + 6x^4 + 9x^3 + 3x^2 - 14x + 1 \]
  \[ c_1 = a_1 \cdot u + 2 \cdot h \]
  \[ = -44x^7 + 15x^6 - 43x^5 + 37x^4 + 37x^3 - 30x^2 - 22x + 42. \]

- The ciphertext is
  
  \((c_0, c_1) = (21x^7 + 2x^6 + 10x^5 + 6x^4 + 9x^3 + 3x^2 - 14x + 1, \]
  \[-44x^7 + 15x^6 - 43x^5 + 37x^4 + 37x^3 - 30x^2 - 22x + 42). \]
Somewhat homomorphic encryption
(Brakerski, Vaikuntanathan 2011)

**SH.Dec**

Given sk = s and a ciphertext ct = (c₀, c₁),

- **compute** \( \tilde{m} = c₀ + c₁s \in \mathbb{R}_q \).

Output the message

- \( \tilde{m} \mod t \).

**Correctness:**

\[
\tilde{m} = c₀ + c₁s = (a₀u + tg + m) + (a₁u + th)s \\
= -(a₁s + te)u + tg + m + a₁us + ths \\
= m + t(g + hs - eu).
\]

Reduction modulo \( t \) gives back \( m \) as long as the error terms are not too large. Gives bound on standard deviation of the Gaussian.
Somewhat homomorphic encryption

Example decryption

- Use sk = s = \(-x^7 - x^6 - x^5 + x^4 + x^3 + x^2 + x - 1\) and ciphertext

\[
(c_0, c_1) = (21x^7 + 2x^6 + 10x^5 + 6x^4 + 9x^3 + 3x^2 - 14x + 1, \\
-44x^7 + 15x^6 - 43x^5 + 37x^4 + 37x^3 - 30x^2 - 22x + 42).
\]

- Compute

\[
\tilde{m} = c_0 + c_1 \cdot s \\
= 24x^7 + 21x^6 + 4x^5 + 21x^4 + 15x^3 + 16x^2 - 28x - 21.
\]

- Reduce modulo \(t = 2\) and get

\[
x^6 + x^4 + x^3 + 1 = m.
\]
Somewhat homomorphic encryption
(Brakerski, Vaikuntanathan 2011)

Homomorphic operations

**SH.Add**
Given $ct = (c_0, c_1)$ and $ct' = (c'_0, c'_1)$, set the new ciphertext

- $ct_{add} = (c_0 + c'_0, c_1 + c'_1)$
- $= (a_0(u + u') + t(g + g') + (m + m'), a_1(u + u') + t(h + h'))$.

**SH.Mult**
Given $ct = (c_0, c_1)$ and $ct' = (c'_0, c'_1)$,

- compute
  
  $$(c_0 + c_1X)(c'_0 + c'_1X) = c_0c'_0 + (c_0c'_1 + c'_0c_1)X + c_1c'_1X^2$$

- $ct_{mlt} = (c_0c'_0, c_0c'_1 + c'_0c_1, c_1c'_1)$

Errors multiply!

$$(m + t(g + hs - eu))(m' + t(g' + h's + eu')) = mm' + t(...)$$
Somewhat homomorphic encryption
(Brakerski, Vaikuntanathan 2011)

- Homomorphic operations increase size of error terms.
- Homomorphic multiplication increases the size of the ciphertext.
- Homomorphic addition, multiplication, and decryption generalize to longer ciphertexts.

**SH.Dec**

Given \( sk = s \) and a ciphertext \( ct = (c_0, c_1, \ldots, c_\delta) \),

- compute \( \tilde{m} = \sum_{i=0}^{\delta} c_i s^i \in R_q \).

Output the message

- \( \tilde{m} \pmod{t} \).
There is a way to go from 3-element ciphertext $ct = (c_0, c_1, c_2)$ back to a 2-element ciphertext.

- We have
  $$c_2 s^2 + c_1 s + c_0 = te_{\text{mult}} + mm'$$

- Publish a “homomorphism key”
  $$h_i = (a_i, b_i = -(a_i s + te_i) + t^i s^2) \quad \text{for } i = 0, \ldots, \lceil \log_t q \rceil - 1$$

- Write $c_2$ in its base-$t$ representation $c_2 = \sum c_{2,i} t^i$. 

Relinearization
(Brakerski, Vaikuntanathan 2011)
Replace $ct$ by $(c_{relin}^0, c_{relin}^1)$ with

$$c_{relin}^1 = c_1 + \sum_{i=0}^{[\log_t q] - 1} c_{2,i} a_i,$$

$$c_{relin}^0 = c_0 + \sum_{i=0}^{[\log_t q] - 1} c_{2,i} b_i.$$

Then

$$c_{relin}^0 + c_{relin}^1 s = c_0 + c_1 s + c_2 s^2 - t e_{relin},$$

$$c_{relin}^0 + c_{relin}^1 s = t (e_{mult} - e_{relin}) + mm'.$$

Okay, ciphertext is smaller, but error has increased!

Decryption still correct if final error $e_{mult} - e_{relin}$ is small enough.
Specific parameter choices

Choosing parameters to “guarantee” security and correctness.

Correctness:

- \( q \) must be large enough when compared to the size of the error terms and \( t \).
- I.e. parameters are chosen s.t. the scheme can evaluate polynomials of a certain fixed degree \( D \) (\( D - 1 \) multiplications and a bunch of additions).

Security:

- Against distinguishing attack with advantage \( 2^{-32} \) by Micciancio/Regev 2009.
- Adjust analysis of Lindner/Peikert 2011 to our setting.
- Still assume RLWE is no easier than LWE.
Specific parameters, key and ciphertext sizes

<table>
<thead>
<tr>
<th>$t$</th>
<th>$D$</th>
<th>$n$</th>
<th>$\lceil \lg(q) \rceil$</th>
<th>$\lg(T)$</th>
<th>$l_{R_q}/10^3$</th>
<th>$(2 + \log_t q) \cdot l_{R_q}/10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>512</td>
<td>19</td>
<td>123</td>
<td>10</td>
<td>205</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1024</td>
<td>38</td>
<td>107</td>
<td>39</td>
<td>1557</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2048</td>
<td>64</td>
<td>134</td>
<td>132</td>
<td>8651</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4096</td>
<td>120</td>
<td>145</td>
<td>492</td>
<td>59966</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>8192</td>
<td>264</td>
<td>117</td>
<td>2163</td>
<td>575276</td>
</tr>
<tr>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>30</td>
<td>164</td>
<td>31</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2048</td>
<td>58</td>
<td>164</td>
<td>119</td>
<td>927</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4096</td>
<td>95</td>
<td>215</td>
<td>390</td>
<td>4475</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>8192</td>
<td>171</td>
<td>242</td>
<td>1401</td>
<td>26756</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>16384</td>
<td>368</td>
<td>214</td>
<td>6030</td>
<td>233938</td>
</tr>
</tbody>
</table>
Message encoding

Homomorphic operations reflect operations in $R_t$.

- Want operations on integers.
- Encode an integer $m = (m_0, m_1, \ldots, m_l)_2$, $m_i \in \{0, 1\}$ as a polynomial of degree $l$ with coefficients $m_i$. Get back $m$ by evaluating at 2.
- $t = 2$ not useful for addition and multiplication since operations mod 2 are different from integer operations.
- Choose $t$ large enough to allow for enough additions.
- Reduction modulo $x^n + 1$ screws up integer multiplication.
- Choose $l$ small enough to allow a certain number of multiplications before reaching degree $n$. 
Reference implementation

Implementation using the computer algebra system Magma
- Uses polynomial arithmetic in Magma,
- no specific optimization for multiplication, no DFT,
- no optimization for specific parameters (sizes),
- decryption for arbitrary length ciphertexts.

Big potential to improve efficiency
- Main cost is polynomial multiplication modulo $x^n + 1$ in $R_q$. 
### Timings

**Intel Core 2 Duo @ 2.1 GHz**

<table>
<thead>
<tr>
<th>$t$</th>
<th>$D$</th>
<th>$n$</th>
<th>$[\lg(q)]$</th>
<th>$S_X$</th>
<th>Enc prec.</th>
<th>Dec deg 1</th>
<th>Dec deg 2</th>
<th>Mult</th>
<th>Mult degred</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>512</td>
<td>19</td>
<td>27</td>
<td>2</td>
<td>2</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>1024</td>
<td>38</td>
<td></td>
<td>55</td>
<td>9</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>2048</td>
<td>64</td>
<td></td>
<td>110</td>
<td>29</td>
<td>18</td>
<td>33</td>
<td>56</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>4096</td>
<td>120</td>
<td></td>
<td>223</td>
<td>85</td>
<td>49</td>
<td>94</td>
<td>163</td>
<td>10.6</td>
</tr>
<tr>
<td>10</td>
<td>8192</td>
<td>264</td>
<td></td>
<td>438</td>
<td>425</td>
<td>227</td>
<td>454</td>
<td>887</td>
<td>114.6</td>
</tr>
<tr>
<td>1024</td>
<td>1</td>
<td>1024</td>
<td>30</td>
<td>54</td>
<td>5</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>2048</td>
<td>58</td>
<td></td>
<td>110</td>
<td>24</td>
<td>15</td>
<td>26</td>
<td>41</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>4096</td>
<td>95</td>
<td></td>
<td>221</td>
<td>81</td>
<td>46</td>
<td>88</td>
<td>154</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>8192</td>
<td>171</td>
<td></td>
<td>440</td>
<td>275</td>
<td>148</td>
<td>288</td>
<td>526</td>
<td>5.3</td>
</tr>
<tr>
<td>10</td>
<td>16384</td>
<td>368</td>
<td></td>
<td>868</td>
<td>1260</td>
<td>664</td>
<td>1300</td>
<td>1593</td>
<td>48.2</td>
</tr>
</tbody>
</table>

- Compute the ciphertext of the sum of 100 numbers of size 128 bits from the single ciphertexts (for mean computation): $< 20\text{ms}$
- Ciphertexts for the sum and sum of squares of 100 such numbers (for mean and variance): $< 6\text{s}$
Questions?

- michael@cryptojedi.org