Parameters for Homomorphic Encryption

Kim Laine and Kristin Lauter

University of California, Berkeley and Microsoft Research

September 3, 2015

Homomorphic Encryption

Consider a public key cryptosystem, and operations $\oplus$, $\otimes$ in ciphertext space, such that

$$\texttt{Encrypt}(m_1) \oplus \texttt{Encrypt}(m_2) = \texttt{Encrypt}(m_1 + m_2)$$
$$\texttt{Encrypt}(m_1) \otimes \texttt{Encrypt}(m_2) = \texttt{Encrypt}(m_1 \cdot m_2)$$

for any plaintexts m_1, m_2. Is this possible to have?

One operation is easy, e.g. RSA:

$$\texttt{RSA-Enc}_e(m_1) = m_1^e \pmod{n}$$

$\texttt{RSA-Enc}_e(m_2) = m_2^e \pmod{n}$

$$\Downarrow$$

$$\texttt{RSA-Enc}_e(m_1 \cdot m_2) = (m_1 \cdot m_2)^e \pmod{n}$$

Or Paillier:

$$\texttt{Pail-Enc}_{\text{pk}}(m_1) \cdot \texttt{Pail-Enc}_{\text{pk}}(m_2) = \texttt{Pail-Enc}_{\text{pk}}(m_1 + m_2)$$

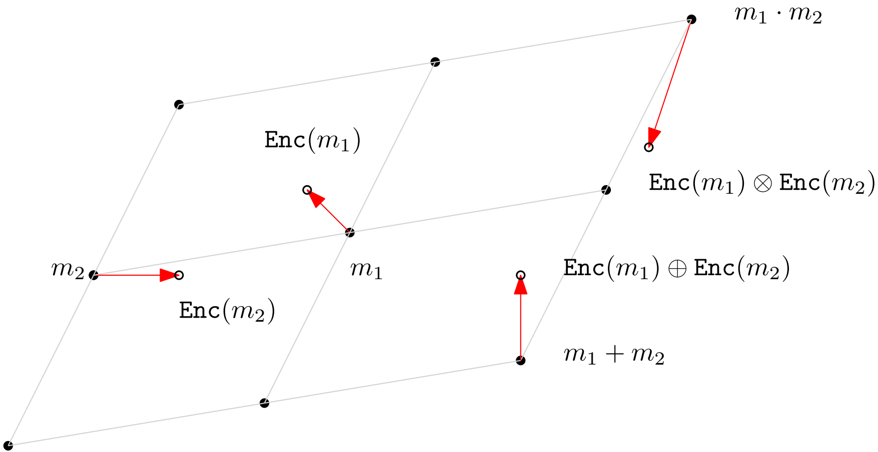
Encryption in lattice-based cryptography:

Messages are encoded as lattice points, and encrypted by adding small displacement (“noise”).

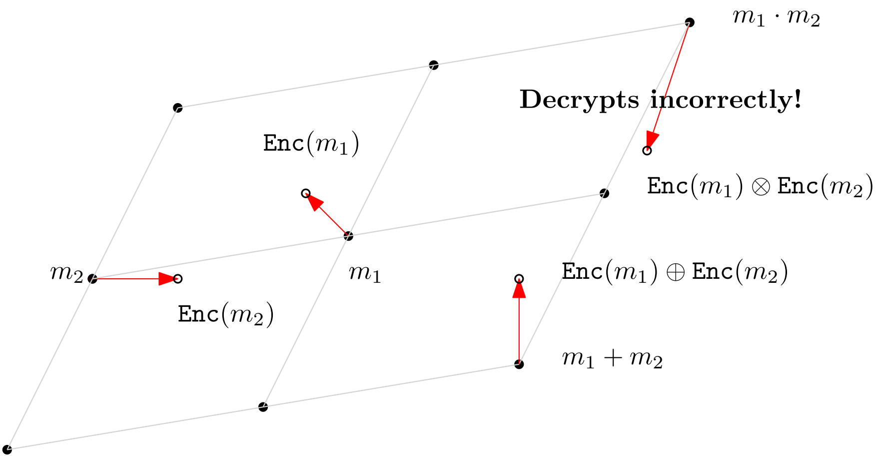
Each fresh ciphertext has an initial noise.

Homomorphic addition:
Noise becomes roughly max of the two input noises.

Homomorphic multiplication:
Noise increases by a multiplicative factor.



Decrypt by recovering the nearest lattice point using secret key information.



Decrypt by recovering the nearest lattice point using secret key information.

Theorem 1 (Bootstrapping Theorem (rough idea))

If the parameters of the cryptosystem are large enough, it is possible to homomorphically decrypt the ciphertext, given an encryption of the secret key, thus refreshing the noise.

Problem: The decryption circuit is typically very deep, so evaluating it requires large parameters.

First steps towards practicality:

1. Encode data to reduce depth of the circuit.
2. Forget about bootstrapping.
3. Select parameters based on the function to be evaluated.
4. Can only do a pre-determined number of homomorphic operations (multiplications)

$\Longrightarrow$ Practical homomorphic encryption

"homomorphic encryption" $\equiv$ "practical homomorphic encryption"

LWE and Ring-LWE

Hard problems for homomorphic encryption:

Learning With Errors (LWE)

- Introduced by Oded Regev
- *On Lattices, Learning With Errors, Random Linear Codes and Cryptography*, 2005

Ring-Learning With Errors (RLWE)

- Introduced by Luybashevsky, Peikert, Regev
- *On Ideal Lattices and Learning With Errors Over Rings*, 2012

LWE and RLWE are closely related lattice problems!

Notation for LWE:

- q an odd prime
- $\mathbf{a}_i, \mathbf{s} \in \mathbb{Z}_q^n$
- $e_j \in \mathbb{Z}_q$ small
- $b_j \in \mathbb{Z}_q$

Learning With Errors:

It is hard to solve secret $\mathbf{s}$ from the linear system

$$\begin{cases} \langle \mathbf{a}_0, \mathbf{s} \rangle + e_0 = b_0 \pmod q \\ \langle \mathbf{a}_1, \mathbf{s} \rangle + e_1 = b_1 \pmod q \\ \langle \mathbf{a}_2, \mathbf{s} \rangle + e_2 = b_2 \pmod q \\ \qquad \vdots \qquad\qquad \vdots \\ \langle \mathbf{a}_{d-1}, \mathbf{s} \rangle + e_{d-1} = b_{d-1} \pmod q \end{cases}$$

unless e_j are known.

Definition 2 (LWE sample)

An LWE sample $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ is one such equation.

Notation for RLWE:

- n a power of 2 (typically 1024, 2048, 4096 or 8192)
- $R_q := \mathbb{Z}_q[x]/(x^n + 1)$
- q an odd prime
- $a_i, s \in R_q$
- $e_j \in R_q$ with small coefficients
- $b_j \in R_q$

Ring-Learning With Errors:

It is hard to solve s from the polynomial system

$$\begin{cases} a_0(x)s(x) + e_0(x) = b_0(x) \\ a_1(x)s(x) + e_1(x) = b_1(x) \\ a_2(x)s(x) + e_2(x) = b_2(x) \\ \qquad \vdots \qquad\qquad \vdots \\ a_{d-1}(x)s(x) + e_{d-1}(x) = b_{d-1}(x) \end{cases}$$

unless $e_j(x)$ are known.

Definition 3 (RLWE sample)

An RLWE sample $(a(x), b(x)) \in R_q \times R_q$ is one such equation.

LWE samples from RLWE samples:

Each RLWE sample will yield one (independent) LWE sample with same parameters by taking the constant coefficient parts.

$$a(x)s(x) + e(x) = b(x)$$

$$\Downarrow$$

$$\underbrace{a[0]s[0] - a[n-1]s[1] - \ldots - a[1]s[n-1]}_{= \langle \mathbf{a}', \mathbf{s} \rangle} + e[0] = b[0] \pmod{q}$$

Error distribution:

The discrete Gaussian distribution with standard deviation σ is a distribution $D_{\mathbb{Z},\sigma}$ on the integers such that

$$\mathrm{Prob}(x) \propto \exp\left(-\frac{x^2}{2\sigma^2}\right) .$$

For security reductions:

In LWE the errors e_i must be sampled from wide enough $D_{\mathbb{Z},\sigma}$, and in RLWE the errors $e_j(x)$ must be sampled coefficient-wise from wide enough $D^n_{\mathbb{Z},\sigma}$.

Brakerski-Vaikuntanathan, CRYPTO 2011:

Setup: Modulus q, $t \geq 2$, n a power of 2, $s \in R_q$
Plaintext space: R_t

Encryption:

Sample $e(x) \leftarrow D^n_{\mathbb{Z},\sigma}$

Sample $a(x) \leftarrow R_q$ uniformly at random

Set $\mathtt{Enc}(m) \leftarrow (a, as + m + te)$

Decryption:

Obtain ciphertext $(a(x), b(x))$

Compute $\mathtt{Dec}(a, b) \leftarrow [b - as] \pmod{t}$

Then $m = \mathtt{Dec}(a, b)$

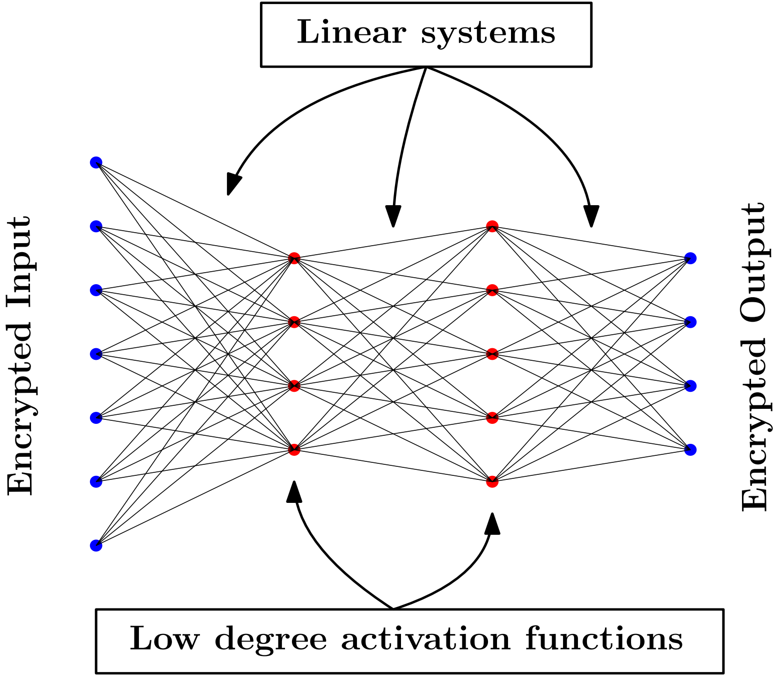
Applications

Predictive analysis of private medical data:

- Predict likelihood of medical conditions from patient's medical data
- Homomorphic encryption guarantees patient privacy
- Bos, Lauter, Naehrig (2013):
  Private Predictive Analysis on Encrypted Medical Data

“Cryptonets”:

- Homomorphic evaluation of suitable neural networks
- Large linear systems are easy to evaluate.
- Activation functions are tricky and need to be carefully chosen.
- Use techniques from cryptography, machine learning, together with special purpose computational tricks to improve efficiency.
- Ongoing joint work with Dowlin, Gilad-Bachrach, Laine, Naehrig, Wernsing


Linear systems
Encrypted Input
Encrypted Output
Low degree activation functions

Predictive analysis of genomic data:

- Genomic data should be considered extremely sensitive.
- From the genome predict the likelihood of traits manifesting in the phenotype (e.g. patient developing Alzheimer's)
- Analysis can be outsourced and performed non-locally, while preserving patient privacy.

Security Properties

Definition 4 (GapSVP (roughly))

Is the shortest vector in a lattice Λ longer than a given gap γ?

Assumption: $\text{GapSVP}_{\gamma(n)}$ is very hard when $\gamma(n) = \text{poly}(n)$.

Theorem 5 (Regev, Peikert (very roughly))

Suppose σ is large enough[1]. Then $GapSVP_{\tilde{O}(nq/\sigma)}$ is easy if LWE is easy.

A similar security result exists for RLWE, but it is more complicated.

[1] Say, bigger than $\sqrt{n}$.

How hard is breaking LWE?

$\text{GapSVP}_{\widetilde{O}(nq/\sigma)}$ gets easier when q increases, other parameters fixed.

No security guarantees for q exponential in n, $\sigma \ll q$.

Theorem 6 (Laine-Lauter)

Any instance of LWE with $q > 2^{2n}$ can be broken in polynomial-time using roughly $2n$ samples. In practice significantly smaller q are vulnerable.

Examples of recovering the LWE secret: $(\sigma = 8/\sqrt{2\pi})$

n	Samples	$\log_2 q$	Time
80	255	16	10m
100	300	19	24m
120	335	22	61m
140	380	24	1.6h
160	420	27	2.9h
180	460	29	4.4h
200	500	32	7.2h
250	600	39	19h
300	705	45	1.8d
350	805	52	3.7d

How is this done?

Consider d samples. Let Λ be the $(n+d)$-dimensional lattice generated by the rows of

$$\begin{bmatrix} q & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & q & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & q & 0 & 0 & \cdots & 0 \\ \mathbf{a}_0[0] & \mathbf{a}_1[0] & \cdots & \mathbf{a}_{d-1}[0] & 1/2^{\ell-1} & 0 & \cdots & 0 \\ \mathbf{a}_0[1] & \mathbf{a}_1[1] & \cdots & \mathbf{a}_{d-1}[1] & 0 & 1/2^{\ell-1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{a}_0[n-1] & \mathbf{a}_1[n-1] & \cdots & \mathbf{a}_{d-1}[n-1] & 0 & 0 & \cdots & 1/2^{\ell-1} \end{bmatrix}$$

Then

$$\mathbf{v} = \left[\langle \mathbf{a}_0, \mathbf{s}\rangle_q, \langle \mathbf{a}_1, \mathbf{s}\rangle_q, \ldots, \langle \mathbf{a}_{d-1}, \mathbf{s}\rangle_q, \mathbf{s}[0]/2^{\ell-1}, \mathbf{s}[1]/2^{\ell-1}, \ldots, \mathbf{s}[n-1]/2^{\ell-1}\right] \in \Lambda$$

$$\mathbf{u} = \left[b_0, b_1, \ldots, b_{d-1}, 0, \ldots, 0\right] \notin \Lambda \text{ but is close to } \mathbf{v} \text{ if } \ell \text{ is big}$$

To recover s:

1. Use LLL to find a reduced basis for Λ.
2. Use Babai's `NearestPlanes` algorithm to find a lattice point close to $\mathbf{u}$.
3. `NearestPlanes` will recover $\mathbf{w} \in \Lambda$ with

$$||\mathbf{w} - \mathbf{u}|| = 2^{\mu(n+d)} \operatorname{dist}(\Lambda, \mathbf{u})$$

   where $\mu \leq 1/4$.
4. But $\mathbf{v}$ is such a lattice point!

How to ensure v is recovered and not some other $\mathbf{w} \in \Lambda$?

Theorem 7 (Laine-Lauter)

If $q > 2^{2n}$ then ℓ and the number of samples can be chosen in such a way that with overwhelming probability the only vector $\mathbf{w} \in \Lambda$ satisfying

$$||\mathbf{w} - \mathbf{u}|| \leq 2^{(n+d)/4} \, dist(\Lambda, \mathbf{u})$$

is $\mathbf{v}$.

In practice $\mu \ll 1/4$ is realized.

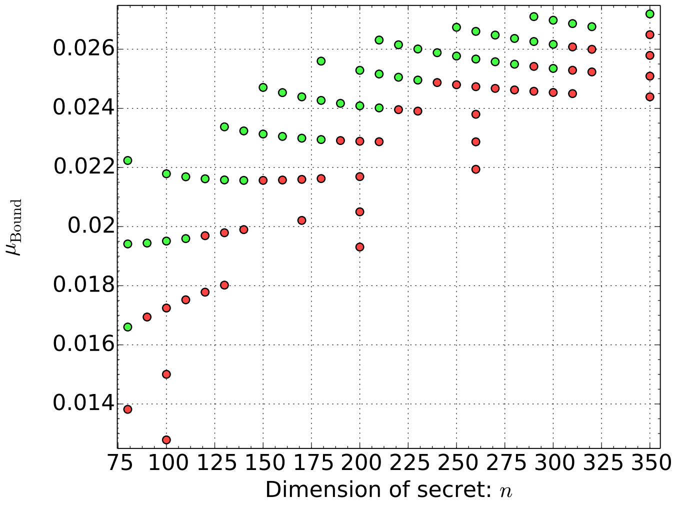
Practical attack:

1. Succeeds almost certainly when (d = number of samples)

$$\mu \leq \mu_{\text{Bound}} := \frac{1}{d} \log_2 \left[\frac{q^{1-n/d}}{2\sigma\sqrt{d}} - 1 \right].$$

2. Choose d in a way that maximizes μ_{Bound}.
3. Run the lattice attack.
4. For security estimates, predict how realized μ is related to the lattice and quality of the basis.

Green dot: Secret recovery succeeded **Red dot:** Secret recovery failed



All experiments were done using SAGE, PARI/GP and fplll.

Open questions:

What happens for larger examples?

What happens if better lattice reduction is used?

Distinguishing attack:

- A direct way of attacking distinguishing problem
- **Laine-Lauter:** Secret recovery becomes easy roughly when distinguishing becomes easy (for same LWE parameters), even without the search-to-decision reduction.
- Success probability depends only on root-Hermite factor (RHF) of basis.

To do:

- Revised LWE security estimates by understanding BKZ-2.0 better?
- How does the special structure of the lattice affect BKZ-2.0 performance?
- How does σ affect hardness of known lattice attacks on secret recovery and distinguishing?