2018-07-16

Nankai-Baidu
Joint Laboratory
Parallel and Distributed
Software Technology Lab

CONTENT

- Somewhat Practical Fully Homomorphic Encryption
  A review of homomorphic encryption and software tools for encrypted statistical machine learning

- Fan and Vercauteren

The scheme of Fan and Vercauteren

Message m must be converted to a polynomial representation $m = m(x)$.

$$m = \sum_{n=0}^{b-1} a_n 2^n \quad \Rightarrow \quad m(x) = \sum_{n=0}^{b-1} a_n x^n$$

Key Generation:

The secret key k_s is simply a uniform random draw from $R_2 \in (-1, 1]$.

(sample a $b = 2^{d-1}$ binary vector for the polynomial coefficients.)

The scheme of Fan and Vercauteren

Key Generation:

The public key k_p is a vector containing two polynomials:

$$k_p = \left(k_{p1}, k_{p2}\right) = (\,[-(a \cdot k_s + e)]_q\,, a)$$

$\text{i.e.}\;\; q = 2^{128} \quad \sigma = 16$

e is a draw from the discrete Gaussian distribution χ, $e \leftarrow \chi$

(defined to be the probability mass function proportional to $e^{-\frac{x^2}{2\sigma^2}}$ over the integers from $-B$ to B, where typically $B \approx 10\sigma$.)

a is uniform random draw from $R_q \in (-q/2, q/2]$.

$[a]_q$ denotes the unique integer in $Z_q = \left\{n : n \in Z, -\frac{q}{2} < n \le \frac{q}{2}\right\}$, which is equal to a mod q.

● The scheme of Fan and Vercauteren

Key Generation:

The public key k_p is a vector containing two polynomials:

$$k_p = \left(k_{p1}, k_{p2}\right) = (\,[-(a \cdot k_s + e)]_q\,, a)$$

$\mathrm{i.e.}\;\; q = 2^{128} \quad \sigma = 16$

e is a draw from the discrete Gaussian distribution χ, $e \leftarrow \chi$

(defined to be the probability mass function proportional to $e^{-\frac{x^2}{2\sigma^2}}$ over the integers from $-B$ to B, where typically $B \approx 10\sigma$.)

a is uniform random draw from $R_q \in (-q/2, q/2]$.

$[a]_q$ denotes the unique integer in $Z_q = \left\{n : n \in Z, -\frac{q}{2} < n \leq \frac{q}{2}\right\}$, which is equal to a mod q.

- The scheme of Fan and Vercauteren

Key Generation:

e is a draw from the bounded discrete Gaussian draw induced on R, χ.

e_j is a scalar discrete random Gaussian draw.

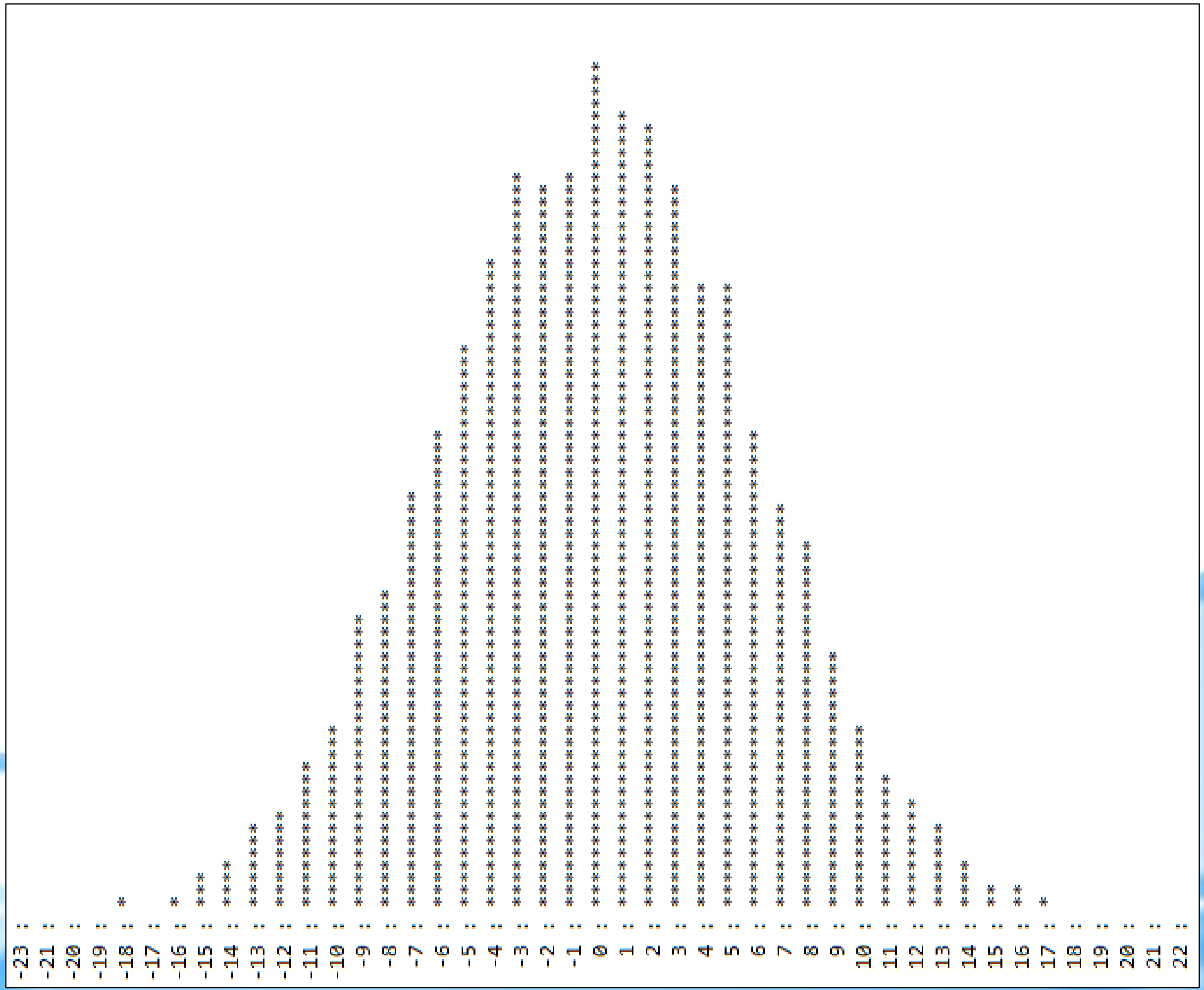
⋯ by a fresh random error term that is relatively concentrated around 0.

e is a draw from the discrete Gaussian distribution χ, $e \leftarrow \chi$

(defined to be the probability mass function proportional to $e^{-\frac{x^2}{2\sigma^2}}$ over the integers from $-B$ to B, where typically $B \approx 10\sigma$.)

e_j是从满足离散高斯分布的整数区间$[-B, B]$中随机抽取的一个整数。

● The scheme of Fan and Vercauteren



a draw from the discrete Gaussian draw over [-10σ , 10σ]

$\mu = 0$

$\sigma = 6$

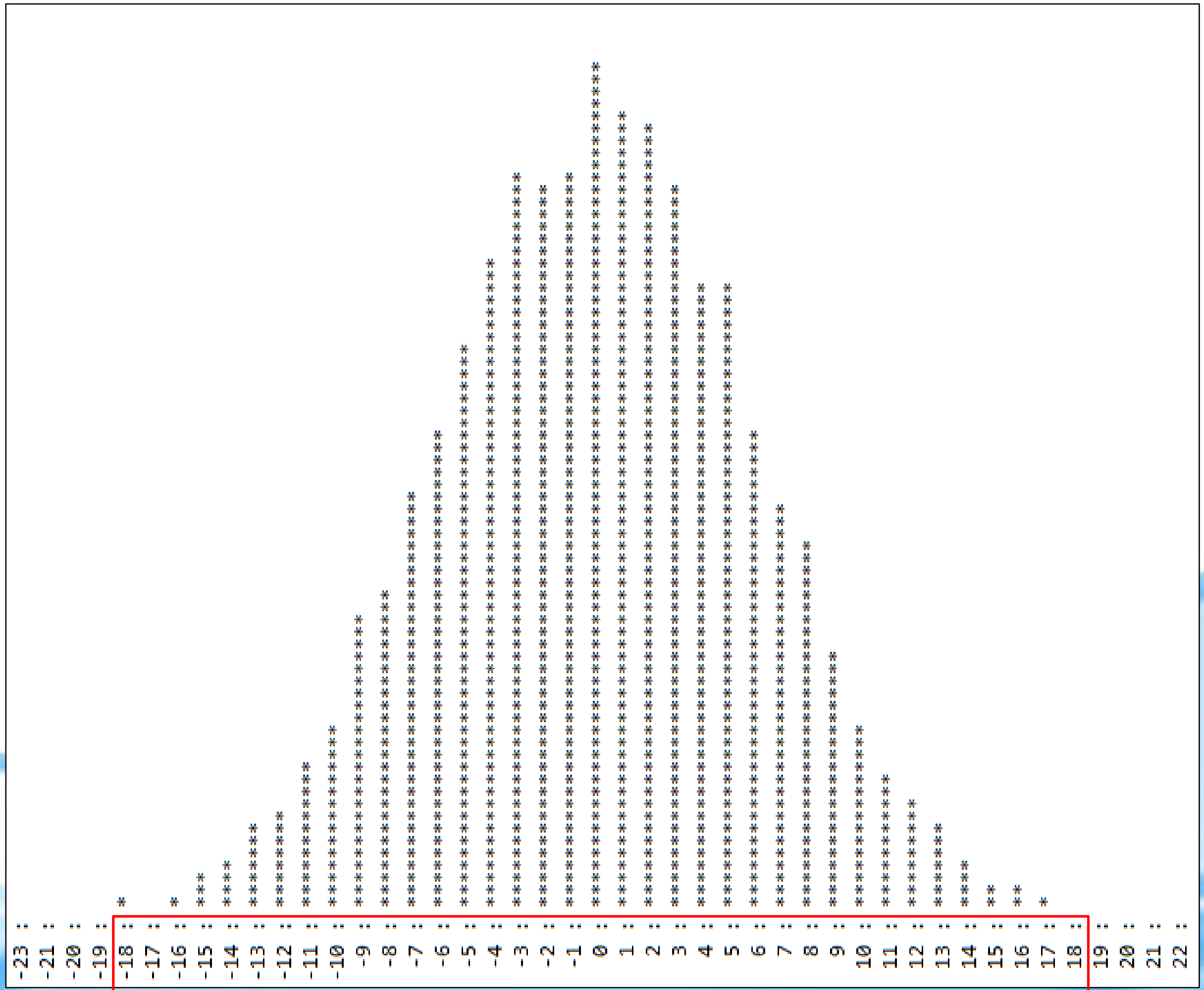
$[\mu-3\sigma , \mu+3\sigma]$

3σ

$P\{|x-\mu| < 3\sigma\} = 2\Phi(3) - 1 \approx 0.9974$

$*$: 10   count : 10000

● The scheme of Fan and Vercauteren



a draw from the discrete Gaussian draw over [-10σ , 10σ]

μ = 0
σ = 6

[μ-3σ , μ+3σ]
3σ
P$\{|x-\mu|<3\sigma\}=2\Phi(3)-1\approx 0.9974$

* : 10 count : 10000

Nankai-Baidu Joint Laboratory

Parallel and Distributed Software Technology Lab

- The scheme of Fan and Vercauteren

Sage Statistics »

previous | next | modules | index

Previous topic

Discrete Gaussian Samplers over the Integers

Next topic

Discrete Gaussian Samplers over Lattices

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

Discrete Gaussian Samplers for $\mathbf{Z}[x]$

This class realizes oracles which returns polynomials in $\mathbf{Z}[x]$ where each coefficient is sampled independently with a probability proportional to $\exp(-(x-c)^2/(2\sigma^2))$.

AUTHORS:

- Martin Albrecht, Robert Fitzpatrick, Daniel Cabracas, Florian Göpfert, Michael Schneider: initial version

EXAMPLES:

```
sage: from sage.stats.distributions.discrete_gaussian_polynomial import DiscreteGaus
sage: sigma = 3.0; n=1000
sage: l = [DiscreteGaussianDistributionPolynomialSampler(ZZ['x'], 64, sigma)() for _ in rang
sage: l = [vector(f).norm().n() for f in l]
sage: mean(l), sqrt(64)*sigma
(23.83..., 24.0...)
```

http://www.sagemath.org/

Rejection Sampling for Discrete Gaussian on Z

Algorithm 1 **Sample**$\mathbb{Z}_m$: Rejection Sampling for Discrete Gaussian on $\mathbb{Z}$

input: A center $t : \mathbb{FP}_m$, and a parameter $\sigma : \mathbb{FP}_m$, and a tailcut parameter $\tau : \mathbb{FP}_m$

output: output $x : \mathbb{Z}$, with distribution statistically close to $D_{\mathbb{Z},t,\sigma}$

1: $h \leftarrow -\pi/\sigma^2 : \mathbb{FP}_m$; $x_{\max} \leftarrow \lceil t + \tau\sigma \rceil : \mathbb{Z}$; $x_{\min} \leftarrow \lfloor t - \tau\sigma \rfloor : \mathbb{Z}$

2: $x \leftarrow$ **RandInt**$(x_{\min}, x_{\max}) : \mathbb{Z}$; $p \leftarrow \exp(h \cdot (x - t)^2) : \mathbb{FP}_m$

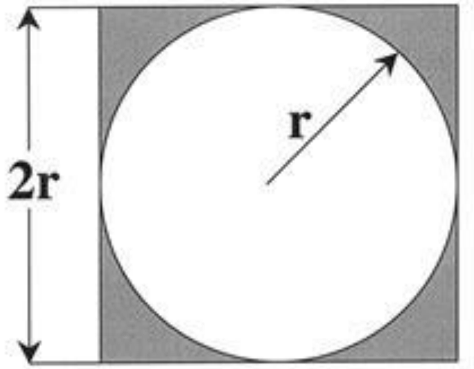
3: $r \leftarrow$ **RandFloat**$_m() : \mathbb{FP}_m$; **if** $r < p$ **then** return x

4: **Goto** Step 2.

✓ Faster Gaussian Lattice Sampling using Lazy Floating-Point Arithmetic

Léo Ducas and Phong Q. Nguyen

- Estimating the value of π



Area of Square = $4r^2$

Area of Circle = πr^2

Ratio of area of Circle to area of Square = $\pi r^2/4r^2$

= $\pi/4$

Total number of throws = N

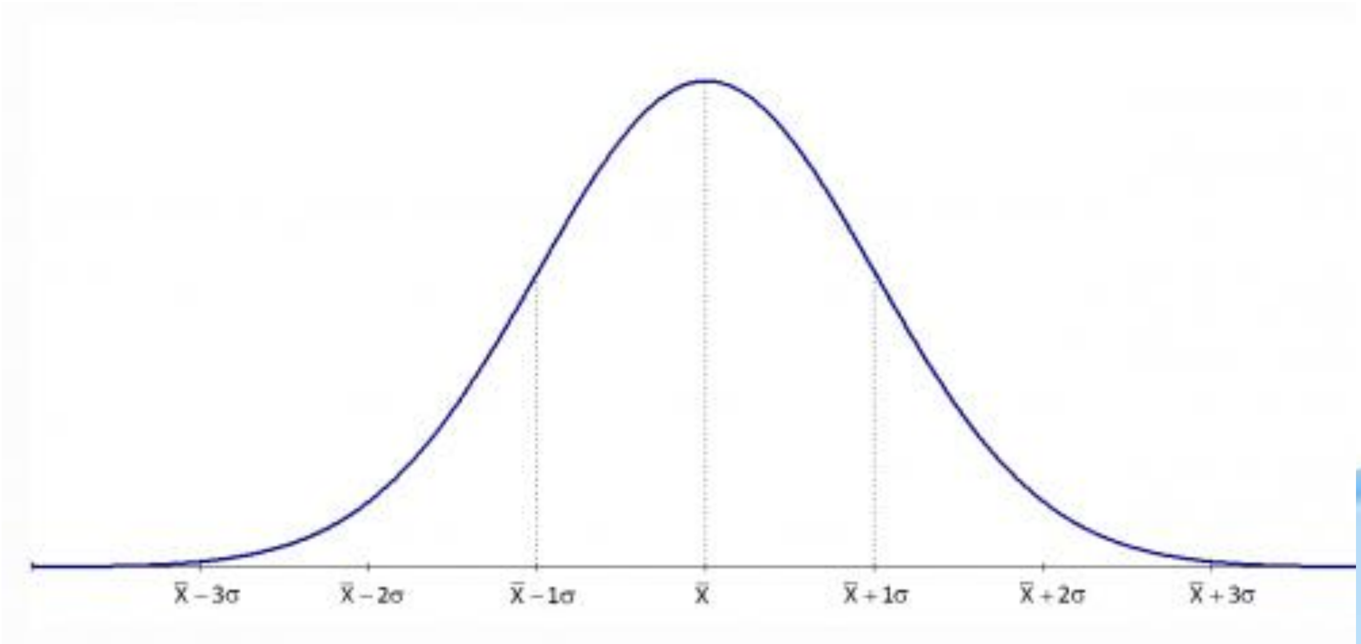
No. hits inside circle = M

Ratio of no. hits inside circle to total no. throws = M/N

$\pi/4 \approx M/N \quad => \quad \pi \approx 4 * M/N$

[Estimating the value of "Pi" by Monte Carlo Methods]

Rejection Sampling for Discrete Gaussian on Z



Step 1. $x \leftarrow [-3\sigma, 3\sigma]$

Step 2. $p \leftarrow \quad ke^{-.5\left(\frac{x-\mu}{\sigma}\right)^2}$

Step 3. $r \leftarrow random(0,1)$

- The scheme of Fan and Vercauteren

✓ FV scheme : Encrypt, Decrypt, Add, Multiply

✓ Next : Turn somewhat HE to Fully HE