An Introduction to Homomorphic Encryption

Jeff Hoffstein
Brown University

20 Years of Cryptography and Security at WPI
October 19, 2015

The situation

Suppose you have a massive amount of data and you would like to perform computations on this data.

There is far more to be done than you have the computational resources to do.

You would like to ship the data off to the cloud, and have these computations done for you.

The trouble is, the data is confidential and whoever or whatever will perform the computation in the cloud is not trusted.

The dream

- encrypt your data
- send off the encrypted data to the cloud
- have the computations done on the encrypted data
- receive the output
- decrypt the output and get the same answer as you would have gotten if the computations had been performed on the unencrypted data

What is needed

First of all, we need an encryption function $e(x)$ with the following properties:

- Given a sequence of inputs $x_1, x_2, \ldots,$ and an encryption key k, it must be easy to compute $e(x_1), e(x_2), \ldots,$
- Given a decryption key d, it must be easy to compute $x_1, x_2, \ldots,$ if one is given $e(x_1), e(x_2), \ldots,$.
- Without knowledge of a decryption key d, it must be hard to compute $x_1, x_2, \ldots,$ if one is given $e(x_1), e(x_2), \ldots$.
- More stringent: Given an arbitrarily long sequence $(x_1, e(x_1)), (x_2, e(x_2)) \ldots$ of plaintext-ciphertext pairs, it must be very hard to obtain any information about a new input x, given a new output $e(x)$.

The homomorphic property

Most importantly, the encryption function must satisfy

$$e(x_1 + x_2) = e(x_1) + e(x_2)$$

and

$$e(x_1 x_2) = e(x_1) e(x_2).$$

In other words, if the data lies in one ring R and the encrypted data lies in another ring S, then

$$e : R \to S$$

must be a ring homomorphism.

Unfortunately, this property is highly incompatible with the security properties.

The problem of 0

Suppose an attacker had the ability to recognize an encryption of 0.

In other words, suppose that if presented with a ciphertext $e(x)$, an attacker could decide whether or not $x = 0$.

In the simplest attack scenario, suppose also that by a lucky guess, or by some small bit of knowledge of the underlying data, an attacker obtains a pair $(1, e(1))$. Then, as $e(2) = e(1) + e(1)$ the pair $(2, e(2))$ is revealed. Similarly, $e(n)$ for any integer n is obtained.

The problem of 0, cont.

Lists of data, although long, are still short enough that for any given $e(x)$, the quantity

$$e(x) - e(n) = e(x - n)$$

can be computed for enough n to find a match $x = n$. As an encryption of 0 can be recognized, this decrypts $e(x)$ and reveals x.

The killer requirement

For this reason the following requirement must be satisfied by any useful homomorphic encryption function:

Given an arbitrarily long sequence of encryptions of 0, and given a new encryption $e(x)$, it must be impractical to answer the question "Does $x = 0$", with a greater than random chance of success.

So actually, e can't be a function. Given any x, there must be a large number of possible $e(x)$.

The ideal recognition problem

Remember that

$$e : R \to S$$

is a ring homomorphism. The decryption function

$$d : S \to R$$

must also be a ring homomorphism, and the collection of all $s \in S$ such that s is an encryption of 0 is exactly the collection of all s such that $d(s) = 0$, which is an ideal.

What's really happening

Think of R as a subring of S. Let I be an ideal of S. Then to encrypt any $x \in R$, choose r at random from I and set $e(x) = x + r$. Then the decryption function d is really a ring homomorphism

$$d : S \to S/I \simeq R.$$

Thus, encryptions of 0 are precisely the elements in I, and the 0 recognition requirement boils down to the following question:

Suppose you have access to an arbitrarily long sequence of elements of I. If you are now given an element r of S, can you determine whether or not $r \in I$.

This is where it gets nasty...

If I is any finite dimensional structure, like a lattice or a vector space then it just takes finitely many $r \in I$ to generate a basis for I.

This forces the dimension of I to be infinite.

It is very tempting to choose a sequence of polynomials in several variables, such as $f_1(x_1, x_2, x_3), f_2(x_1, x_2, x_3), \ldots, f_n(x_1, x_2, x_3)$, and to let I be the ideal in, say, $\mathbb{Z}/q\mathbb{Z}[x_1, x_2, x_3]$ generated by the f_i.

The attempt

If you have secret knowledge of solutions to

$$f_1(x_1, x_2, x_3) = f_2(x_1, x_2, x_3) = \cdots = f_n(x_1, x_2, x_3) = 0$$

then the substitution of these values into an $f \in I$ sends f to 0.

If we think of $R = \mathbb{Z}/q\mathbb{Z}$ as a subring of $\mathbb{Z}/q\mathbb{Z}[x_1, x_2, x_3]$, and an encryption of $n \in \mathbb{Z}/q\mathbb{Z}$ as

$$e(n) = n + f(x_1, x_2, x_3),$$

for a random choice of $f \in I$, then substituting the secret solution recovers n.

What goes wrong

If the number of variables is at all large there is an explosion of the size of ciphertexts as encrypted data is multiplied.

Many monomials of the form $x_1^a x_2^b x_3^c \ldots$ are formed.

If the number of variables is not large, then the method of Gröbner bases has proved to be remarkably successful in identifying I.

The advent of noise

In 2009, Craig Gentry came up with a new idea: Don't use a perfect ring homomorphism.

Suppose that instead of encrypting 0 as $r \in I$ for a random choice of r in the ideal I, one encrypted 0 as $r + \epsilon$.

To decrypt, first reduce mod I to eliminate the r, and then do some kind of "rounding" to get rid of ϵ.

The ϵ destroys the algebraic structure, but if it is small enough it can be eliminated by the decryptor.

The advent of noise, cont.

The problem is that when multiplying encrypted data one has

$$(m_1 + r_1 + \epsilon_1)(m_2 + r_2 + \epsilon_2)$$

$$= m_1 m_2 + m_1 r_2 + m_1 \epsilon_2 + r_1 m_2 + r_1 r_2 + r_1 \epsilon_2 + \epsilon_1 m_2 + \epsilon_1 r_2 + \epsilon_1 \epsilon_2.$$

The extra terms

$$m_1 \epsilon_2 + r_1 \epsilon_2 + \epsilon_1 m_2 + \epsilon_1 r_2 + \epsilon_1 \epsilon_2$$

seriously impact the noise level.

Bootstrapping

Gentry's solution was a recursive one in which the data processing program would, after a few operations, decrypt its output homomorphically using a homomorphic encryption of a decryption key.

This accomplished a theoretical breakthrough in which a partially homomorphic scheme was leveraged into a fully homomorphic scheme, but this advance came with a cost of highly increased complexity.

Subsequent work

Gentry's first proposal was inspired by the ring-based NTRU public key cryptosystem introduced in 1996 by H, Pipher,J. and Silverman, J.

In this system, encryption is partially homomorphic in the sense that with a proper choice of parameters a very small number of multiplications and many additions can be applied to encrypted data, and can be decrypted homorphically to the images of the same operations performed on the original data.

Three significant recent improvements:

- The Brakerski-Gentry-Vaikuntanathan cryptosystem (BGV),
- The NTRU-based cryptosystem due to Lopez-Alt, Tromer, and Vaikuntanathan (LTV) (Improved significantly in 2014 by Dai, Doröz, and Sunar.)
- The Gentry-Sahai-Waters cryptosystem (GSW).

The rings appearing in NTRU

NTRU uses the rings $\mathbb{Z}[x]/(x^N-1)$ and $\mathbb{Z}/q\mathbb{Z}[x]/(x^N-1)$. Here multiplication works as follows: If

$$f(x)=\sum_{i=0}^{N-1} a_i x^i \quad \text{and} \quad g(x)=\sum_{i=0}^{N-1} b_i x^i,$$

Then

$$f*g := f(x)g(x) \equiv \sum_{k=0}^{N-1} c_k x^k \pmod{x^N-1},$$

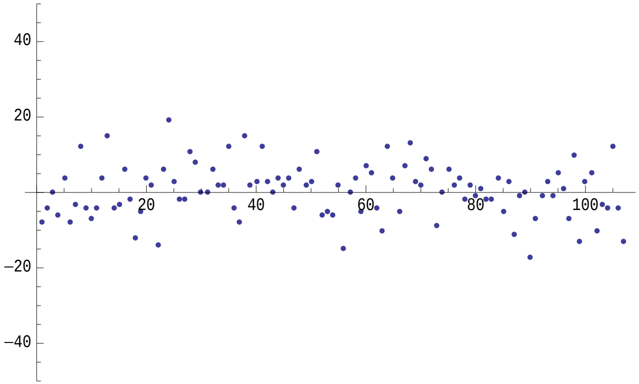
where

$$c_k = \sum_{i+j\equiv 0 \pmod{N}} a_i b_j.$$

Short times short equals short

For example, take $N = 107$, and f, g to have coefficients from $\{-1, 0, 1\}$. Then the distribution of coefficients of $f * g$ looks like

Out[26]=



The underlying idea

Suppose

$$r(x) = r_1(x) + 3r_2(x),\ \ s(x) = s_1(x) + 3s_2(x) \in \mathbb{Z}/q\mathbb{Z}[x]$$

where $r_1(x), r_2(x), s_1(x)$ and $s_2(x)$ have small coefficients, say, chosen from $\{1, 0, -1\}$.

Then $r(x)s(x) \pmod{x^N - 1, q}$ will still have small coefficients with absolute value less than $q/2$.

As a result it makes sense to further reduce $r(x)s(x)$ mod 3, after reducing mod q, so

$$r_1(x)s_1(x) \equiv \Big(r(x)s(x) \pmod{x^N - 1, q}\Big) \pmod{3}.$$

The underlying idea, cont.

This is the main idea underlying NTRU, and it is also the underlying principle behind most of the homomorphic schemes proposed to date.

The point is that 0 is encrypted as $pr(x)$ for any random choice of $r(x)$ with small coefficients. A number m is encrypted as $m + pr(x)$. The $pr(x)$ is the "noise", and this is eliminated by reduction mod p following a reduction mod q.

Proposed approach by H, Silverman and Sunar

Instead of $\mathbb{Z}/q\mathbb{Z}[x]/(x^N - 1)$, take q to be prime and use rings such as $R = \mathbb{Z}/q\mathbb{Z}[x]/(f(x))$ where, say

$$f(x) = x^N - x^{N-2} + \cdots + x^2 - 1$$

is an irreducible polynomial modulo q with small coefficients.

The key point: As long as $f(x)$ has small coefficients It is *still* true in R that short times short equals short.

Another key point: R is now a finite field of order q^N. All finite fields of the same order are isomorphic, so if $g(y)$ is any other degree N polynomial that is irreducible over q, then

$$\mathbb{Z}/q\mathbb{Z}[y]/(g(y)) \simeq \mathbb{Z}/q\mathbb{Z}[x]/(f(x))$$

Proposed approach, cont.

Public working space is $\mathbb{Z}/q\mathbb{Z}[y]/(g(y))$, private working space is $\mathbb{Z}/q\mathbb{Z}[x]/(f(x))$

These are connected by a secret isomorphism:

$$x \to \varphi(y) \pmod{g(y), q}.$$

That is, encryption is via the isomorphism that sends

$$e(m(x) + pr(x)) \equiv m(\varphi(y)) + pr(\varphi(y)) \pmod{g(y), q}.$$

Here $\varphi(y)$ is a secret polynomial in y of degree $\leq N$.

The main point

The isomorphism between $\mathbb{Z}/q\mathbb{Z}[y]/(g(y))$, *and* $\mathbb{Z}/q\mathbb{Z}[x]/(f(x))$ *does not respect the Archimedian property of size. The image of a short polynomial in* x *is a polynomial in* y *with coefficients uniformly distributed mod* q.

The main point, cont.

Parameters and number of multiplications are set so that if the arithmetic were done in $\mathbb{Z}/q\mathbb{Z}[x]/(f(x))$ the output would be polynomials with coefficients having absolute value less than $q/2$. Thus noise can be eliminated by reduction first mod q and then mod 3.

The arithmetic will be done by the cloud in $\mathbb{Z}/q\mathbb{Z}[y]/(g(y))$ and coefficients will appear to be randomly distributed mod q.

To decrypt, the user applies the secret isomorphism, recovering, after reduction mod q, a polynomial in x with short coefficients.

Finally, the user reduces mod p recovering the output of the calculation, with the noise removed.

Attacks

To decrypt, an attacker must find a field isomorphism that sends each encrypted piece of data, taking the form of a polynomial in y with coefficients uniformly distributed mod q, into a new field where all the coefficients are short.

All known attacks of this sort are done via lattice reduction, and involve a search for a linear transformation of vector spaces.

There are many potential solutions to this, but with high probability there will a unique linear transformation that is also a field isomorphism.

Only a field isomorphism will be useful for decryption.

Concluding point

Lattice reduction attacks force current homomorphic schemes to take N in the thousands.

Because of the fundamental *non-linearity* of this scheme, it appears to be possible to take N far smaller, in the low hundreds.

Our belief is that this will vastly increase the efficiency of our scheme . . .

. . . but this our dream, and not a fact. Lots of research needs to be done!

Thanks!!

;