Let's revisit Regev's encryption scheme. It turns out that it readily generalizes to give a <u>fully</u> homomorphic encryption scheme.

<u>Abstractly</u>: given encryption ct_x of value x under some public key, can we derive from that an encryption of $f(x)$ for an arbitrary function f?

- So far, we have seen examples of encryption schemes that support <u>one</u> type of operation (e.g., addition) on ciphertexts
  - ElGamal encryption (in the exponent): homomorphic with respect to addition
  - Boneh-Goh-Nissim: addition + 1 multiplication
- For FHE, need homomorphism with respect to two operations: addition and multiplication

  <u>Major open problem</u> in cryptography (dates back to late 1970s!) — first solved by Stanford student Craig Gentry in 2009

  ↳ revolutionized lattice-based cryptography!

  ↳ very surprising this is possible: encryption needs to "scramble" messages to be secure, but homomorphism requires preserving <u>structure</u> to enable arbitrary computation

<u>General blueprint</u>: 1. Build somewhat homomorphic encryption (SWHE) — encryption scheme that supports <u>bounded</u> number of homomorphic operations

2. Bootstrap SWHE to FHE (essentially a way to "refresh" ciphertext)

Focus will be on building SWHE (has all of the ingredients for realizing FHE)

<u>Starting point</u>: Regev encryption

$$\left.\begin{aligned} &\text{pk:} \quad A = \begin{bmatrix} \bar{A} \\ \bar{s}^T\bar{A} + e^T \end{bmatrix} \in \mathbb{Z}_q^{n \times m} \\ &\text{sk:} \quad s^T = [-\bar{s}^T \mid 1] \in \mathbb{Z}_q^n \end{aligned}\right\} \text{Invariant: } s^TA = e^T$$

$$\text{ct:} \quad r \xleftarrow{R} \{0,1\}^m, \quad c \leftarrow Ar + \begin{bmatrix} 0^{n-1} \\ \lfloor q/2 \rceil \cdot \mu \end{bmatrix} \in \mathbb{Z}_q^n$$

$$\hookrightarrow s^Tc = s^T\left(Ar + \begin{bmatrix} 0^{n-1} \\ \lfloor q/2 \rceil \cdot \mu \end{bmatrix}\right) = e^Tr + \lfloor q/2 \cdot \mu \rceil.$$

as long as e^Tr is small, decryption succeeds

Essentially, with Regev encryption, the decryption invariant if

$$s^Tc = \mu \cdot \lfloor \tfrac{q}{2} \rfloor + \text{error}$$

Suppose however that instead of encrypting μ, we encrypted the entries of $\mu \cdot s^T$ instead. And also ignore the scaling factor. Then, the ciphertext would be a matrix $C \in \mathbb{Z}_q^{n \times n}$ where

$$s^TC = \mu \cdot s^T + \text{error} \in \mathbb{Z}_q^n$$

↳ specifically: $C = AR + \mu \cdot I_n$ ⟶ $s^TC = \overset{s^TA = e^T}{s^TAR} + \mu \cdot s^T$

where $R \xleftarrow{R} \{0,1\}^{m \times n}$

$$= \underbrace{e^TR}_{\text{error}} + \mu \cdot s^T$$

Observe: Suppose C_1 was a Regev encryption of $\mu_1 \cdot s^T$ and C_2 was Regev encryption of $\mu_2 \cdot s^T$. Then:

$$s^TC_1C_2 = (\mu_1 \cdot s^T + e_1^T)C_2 = \mu_1(\mu_2 \cdot s^T + e_2^T) + e_1^TC_2$$

$$= \mu_1\mu_2 \cdot s^T + \mu_1 e_2^T + e_1^TC_2$$

This is basically an encryption of $\mu_1\mu_2$ with new error term $\mu_1 e_2^T + e_1^T C_2$.

small since $\mu_1 \in \{0,1\}$ and e_2^T is small

big because C_2 is a Regev ciphertext (has large entries over $\mathbb{Z}_q^{n\times n}$)

Due to the large noise, <u>cannot</u> recover the message anymore...
Need a way to avoid multiplying by something large.

- How to make something small? Binary decomposition!

First, we define the "gadget" matrix (there are actually many possible gadget matrices — here, we use a common one sometimes called the "powers-of-two" matrix):

$$G = \begin{pmatrix} 1 \; 2 \; 4 \; 8 \; \cdots \; 2^{\lceil \log q \rceil - 1} & & & \\ & 1 \; 2 \; 4 \; \cdots \; 2^{\lceil \log q \rceil - 1} & & \\ & & \ddots & \\ & & & 1 \; 2 \; 4 \; \cdots \; 2^{\lceil \log q \rceil - 1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 2 & 4 & \cdots & 2^{\lceil \log q \rceil - 1} \end{pmatrix}}_{g^T} \otimes I_n = g^T \otimes I_n$$

Each row of G consists of the powers of two (up to $2^{\lceil \log q \rceil - 1}$). Thus, $G \in \mathbb{Z}_q^{n \times n\lceil \log q \rceil}$. Oftentimes, we will just write $G \in \mathbb{Z}_q^{n \times m}$ where $m > n\lceil \log q \rceil$. Note that we can always pad G with all-zero columns to obtain the desired dimension.

<u>Observation</u>: given any $y \in \mathbb{Z}_q^n$, it is easy to find an $x \in \{0,1\}^m$ where $Gx = y$.

Let $y_{i,\lceil \log q \rceil - 1}, \ldots, y_{i,0}$ be the binary decomposition of y_i (the i^{th} component of y). Then,

$$G \cdot \begin{pmatrix} y_{1,0} \\ y_{1,2} \\ \vdots \\ y_{1,\lceil \log q \rceil - 1} \\ y_{2,0} \\ \vdots \\ y_{2,\lceil \log q \rceil - 1} \\ \vdots \\ y_{n,0} \\ \vdots \\ y_{n,\lceil \log q \rceil - 1} \end{pmatrix} = \begin{pmatrix} \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j y_{1,j} \\ \vdots \\ \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j y_{n,j} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = y$$

Observe that this is a 0/1 vector (binary valued vector)

We will denote this "bit-decomposition" operation by the function $G^{-1} : \mathbb{Z}_q^n \to \{0,1\}^m$

important: G^{-1} is <u>not</u> a matrix (even though G is)!

Then, for all $y \in \mathbb{Z}_q^n$, $G \cdot G^{-1}(y) = y$ and $\|G^{-1}(y)\| = 1$.

ℓ_∞-norm (max absolute value of component of the vector)

Returning to FHE:

Approach: instead of encrypting $\mu \cdot s^T$, we will encrypt $\mu \cdot s^T G$ instead.

Invariant: C is an encryption of μ if

$$s^T C = \mu \cdot s^T G \in \mathbb{Z}_q^m$$

We can construct C as

$$C = AR + \mu G \in \mathbb{Z}_q^{n \times m}.$$

Then $s^T C = s^T AR + \mu \cdot s^T G = e^T R + \mu \cdot s^T G$

Suppose we have two ciphertexts C_1 and C_2 where

$$s^T C_1 = \mu_1 \cdot s^T G + e_1^T$$
$$s^T C_2 = \mu_2 \cdot s^T G + e_2^T$$

Then $C_1 + C_2$ is an encryption of $\mu_1 + \mu_2$:

$$s^T (C_1 + C_2) = (\mu_1 + \mu_2) \cdot s^T G + e_1^T + e_2^T \qquad \text{[erros add]}$$

To multiply, we compute $C_1 G^{-1}(C_2)$:

$$\begin{aligned} s^T C_1 G^{-1}(C_2) &= (\mu_1 \cdot s^T G + e_1^T) G^{-1}(C_2) \\ &= \mu_1 \cdot s^T C_2 + e_1^T G^{-1}(C_2) \\ &= \mu_1 \mu_2 \cdot s^T G + \underbrace{\mu_1 e_2^T + e_1^T G^{-1}(c_2)}_{\text{small since } \mu_1 e_2^T + e_1^T G^{-1}(C_2) \text{ is also small}} \end{aligned}$$

To decrypt a ciphertext C, can compute $s^T C \cdot G^{-1}(\lfloor \frac{q}{2} \rceil \cdot u_n)$ where $u_n = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$ since $s^T u_n = 1$.
As long as total error is less than $\frac{q}{4}$, decryption recovers message

This gives the Gentry-Sahai-Waters encryption scheme.

- $\text{Setup}(1^\lambda)$: Sample $\bar{A} \xleftarrow{R} \mathbb{Z}_q^{(n-1) \times m}$, $\bar{s} \xleftarrow{R} \mathbb{Z}_q^{n-1}$, $e \leftarrow \chi^m$ $\longrightarrow$ $pk = A = \begin{bmatrix} \bar{A} \\ \bar{s}^T \bar{A} + e^T \end{bmatrix}$, $sk = s = [-\bar{s} \mid 1]$ $\qquad (s^T A = e^T)$
- $\text{Encrypt}(A, \mu)$: $R \xleftarrow{R} \{0,1\}^{m \times m}$
  $C \leftarrow AR + \mu \cdot G \in \mathbb{Z}_q^{n \times m}$
- $\text{Decrypt}(s, C)$: compute $s^T C G^{-1}(\frac{q}{2} \cdot I_n)$ and round as usual

Security is *same* argument as for Regev encryption:
Namely, by LWE, the public key is indistinguishable from a uniformly *random* matrix $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$
by LHL, (A, AR) is indistinguishable from (A, U) where $U \xleftarrow{R} \mathbb{Z}_q^{n \times m}$

$\Rightarrow$ $U + \mu G$ perfectly hides μ.

Let's look at noise growth. Suppose $C_1 = AR_1 + \mu_1 G$

$C_2 = AR_2 + \mu_2 G$

Then $s^T C_1 = s^T AR_1 + \mu_1 s^T G = \mu_1 + s^T G + e^T R_1$

$\uparrow$ noise in the ciphertext : must be small relative to q in order to decrypt

Noise increases with each operation:

$$C_1 + C_2 = A(R_1 + R_2) + (\mu_1 + \mu_2) G \quad \rightsquigarrow \text{ new noise is } R_1 + R_2$$

$$\begin{aligned} C_1 G^{-1}(C_2) &= AR_1 G^{-1}(C_2) + \mu_1 C_2 \\ &= A(R_1 G^{-1}(C_2) + \mu_1 R_2) + \mu_1 \mu_2 G \quad \rightsquigarrow \text{ new noise is } R_1 G^{-1}(C_2) + \mu_1 R_2 \end{aligned}$$

norm is bounded by $\|R_1\|_\infty \cdot m + \|R_2\|_\infty$ when $\mu_1 \in \{0,1\}$.

After computing d repeated squarings: noise is $m^{O(d)}$. Will eventually overwhelm q. Thus, there is a bound on number of homomorphic operations the scheme supports.

Fully homomorphic encryption: support arbitrary number of computations.

<u>From SWHE to FHE</u>. The above construction requires imposing an a priori bound on the multiplicative depth of the computation. To obtain fully homomorphic encryption, we apply Gentry's brilliant insight of bootstrapping.

<u>High-level idea</u>. Suppose we have SWHE with following properties:

1. We can evaluate functions with multiplicative depth d
2. The decryption function can be implemented by a circuit with multiplicative depth $d' < d$

Then, we can build an FHE scheme as follows:

- Public key of FHE scheme is public key of SWHE scheme <u>and</u> an encryption of the SWHE decryption key under the SWHE public key
- We now describe a ciphertext-refreshing procedure:
  - For each SWHE ciphertext, we can associate a "noise" level that keeps track of how many more homomorphic operations can be performed on the ciphertext (while maintaining correctness).
    - ↳ for instance, we can evaluate depth-d circuits on fresh ciphertexts; after evaluating a single multiplication, we can only evaluate circuits of depth-$(d-1)$ and so on ...
  - The refresh procedure takes any valid ciphertext and produces one that supports depth-$(d-d')$ homomorphism; since $d > d'$, this enables <u>unbounded</u> (i.e., arbitrary) computations on ciphertexts

<u>Idea</u>: Suppose we have a ciphertext ct where $\text{Decrypt}(sk, ct) = x$.

To refresh the ciphertext, we define the Boolean circuit $C_{ct} : \{0,1\}^{n \log q} \to \{0,1\}$ where $C_{ct}(sk) := \text{Decrypt}(sk, ct)$ and homomorphically evaluate C_{ct} on the encryption of sk

↳ $\text{Encrypt}(pk, sk) \to \text{Encrypt}(pk, C_{ct}(sk))$

$$\text{Encrypt}(pk, C_{ct}(sk)) = x$$

fresh ciphertext that supports d levels

homomorphic evaluation consumes d' levels

refreshed ciphertext still supports $d-d'$ levels of multiplication

Security now requires that the public key includes a copy of the decryption key

↳ Requires making a "circular security" assumption

<u>Open question</u>: FHE <u>without</u> circular security from LWE (possible from iO)

Can be shown that GSW is bootstrappable [Decryption operation is linear, followed by rounding — can be implemented with low-depth circuit]