Addition: $C_1 + C_2$ is encryption of $\mu_1 + \mu_2$:

$$C_1 + C_2 = A(R_1 + R_2) + (\mu_1 + \mu_2)\cdot G$$

New error: $R_+ = R_1 + R_2$, $\|R_+\|_\infty \le \|R_1\|_\infty + \|R_2\|_\infty$

Multiplication: $C_1 G^{-1}(C_2)$ is encryption of $\mu_1 \cdot \mu_2$:

$$\begin{aligned} C_1 G^{-1}(C_2) &= (AR_1 + \mu_1 G) G^{-1}(C_2) \\ &= AR_1 G^{-1}(C_2) + \mu_1 G \cdot G^{-1}(C_2) \\ &= AR_1 G^{-1}(C_2) + \mu_1 C_2 \\ &= AR_1 G^{-1}(C_2) + \mu_1 (AR_2 + \mu_2 G) \\ &= A\underbrace{(R_1 G^{-1}(C_2) + \mu_1 R_2)}_{R_\times} + \mu_1\mu_2 G \end{aligned}$$

New error: $R_\times = R_1 G^{-1}(C_2) + \mu_1 R_2$, $\|R_\times\|_\infty \le \|R_1\|_\infty \cdot m + \|R_2\|_\infty$

After computing d repeated squarings: noise is $m^{O(d)}$

[for correctness, require that $q > 4mB \cdot \|R\|_\infty$, so bit-length of q scales with multiplicative depth of circuit
↳ also requires super-poly modulus when $d = \omega(1)$ (stronger assumption needed)]

But not quite fully homomorphic encryption: we need a bound on the (multiplicative) depth of the computation

<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

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

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

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)

Let's take a closer look at bootstrapping for GSW encryption:

$pk: A \in \mathbb{Z}_q^{n \times m}$ $\quad$ $\mathrm{Enc}(pk, \mu): C \leftarrow AR + \mu \cdot G$

$sk: s \in \mathbb{Z}_q^n$ $\quad$ $\mathrm{Dec}(sk, c)$: compute $s^T C$ and round

specifically: $s^T C \approx \mu \cdot s^T G$

$\left[\begin{matrix} 1 \; 2 \cdots \frac{q}{2} & & \\ & \ddots & \\ & & 1 \; 2 \cdots \frac{q}{2} \end{matrix} \right]$ (when q is power of two)

↳ similar analysis applies for non-power-of-2

let c_m be last column of C

$\Rightarrow \mu \cdot s^T c_m = \mu [-\tilde{s}^T | 1]\, c_m = \mu \cdot \frac{q}{2}$

[recall: $s^T C = s^T AR + \mu\, s^T G = e^T R + \mu \cdot s^T G$]

Consider a computation with multiplicative depth d: can support by setting $q > m^{O(d)}$

Consider depth of circuit implementing GSW decryption: circuit has ciphertext column $c_m \in \mathbb{Z}_q^n$ hard-wired and takes secret key $s \in \mathbb{Z}_q^n$ as input

Need to compute $\mathrm{round}(s^T c_m \bmod q)$ as a Boolean circuit:

- We can write

$$s^T c_m = \sum_{i=1}^{n} \sum_{j=0}^{\lfloor \log q \rfloor} s_{ij} \cdot (2^j \cdot c_{m,i} \bmod q)$$

j^{th} bit of i^{th} component of s

left shift of the binary representation

i^{th} component of c_m (available in the clear)

multiplication by 1 bit = AND gate

Computing $s^T c_m$ over the integers can be computed by $O(n \log q)$ additions of values with $O(\log n + \log q)$ bits

Using an addition tree, this can be computed by a circuit of depth $O(\log n + \log\log q)$ ← need to be careful since adding 2 k-bit values requires circuit of depth $O(\log k)$, but can use a "3→2 trick" to add n k-bit values in depth $O(\log n + \log k)$.

- Given $s^T c_m$ over the <u>integers</u>, need to reduce mod q
  ↳ Can do this brute force: $|s^T c_m| \leq n \log q \cdot q$, so need to subtract by at most $n \log q$ multiples of q
  sum only has $n \log q$ terms
  ↳ Compute all possible multiples of q and <u>select</u> for the one that is within $\mathbb{Z}_q$
    ↳ Selection is tree of AND gates, computable in depth $O(\log n + \log\log q)$

- Recovering 0/1 from $s^T c_m \pmod q$ is just rounding (checking most significant bits of binary representation — constant depth)

Overall depth: $O(\log n + \log\log q) = O(\log n)$ since we always have $q < 2^n$ (for security).

To bootstrap, it suffices to support multiplicative depth $O(\log n)$.
For correctness, we thus require that $q \sim m^{O(\log n)}$, so this is easily satisfiable!

$\Rightarrow$ FHE from LWE + circular security

But... we did require super-polynomial modulus for correctness: $q > m^{O(\log n)}$.
↳ Hardness based on worst-case lattice problems with <u>super-polynomial</u> approximation factor — stronger assumption than for PKE

recall approximation factor based on modulus-to-noise ratio

Can do better by relying on asymmetric noise growth of GSW multiplication:

$C_1 = AR_1 + \mu_1 G$
$C_2 = AR_2 + \mu_2 G$

$$\Rightarrow C_\times = C_1 G^{-1}(C_2)$$
$$= A\underbrace{(R_1 G^{-1}(C_2) + \mu_1 R_2)}_{R_\times} + \mu_1\mu_2 G$$

$\|R_\times\|_\infty \leq \|R_1\|_\infty \cdot m + \|R_2\|_\infty$

<u>Observe</u>: $R_\times$ only scales R_1, dependence on R_2 is <u>additive</u>

Suppose we have $C_1, \ldots, C_t$ with noise $R_1, \ldots, R_t$ where $\|R_i\|_\infty \leq B$ for all $i \in [t]$.
Consider sequence of homomorphic multiplications where each multiplication involves one of $C_1, \ldots, C_t$. Then, noise accumulation after T multiplications is bounded by $T \cdot B \cdot m$

each multiplication increases noise by <u>additive</u> factor $B \cdot m$

<u>Key takeaway</u>: if input to every multiplication is a <u>fresh</u> ciphertext, then noise growth is <u>additive</u> not multiplicative in the depth

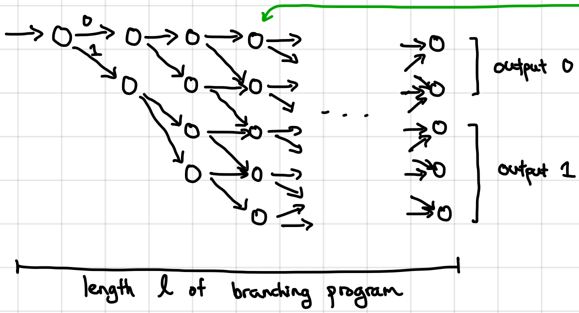
Asymmetric noise growth extremely useful both theoretically and practically!

← very efficient private information retrieval protocols

← base security on weaker assumptions ($\approx$ PKE!)

How to exploit in the case of bootstrapping? Rounded inner product does not necessarily have this form...

Branching programs: one way to capture space-bounded computations

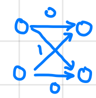


State can be expressed as an indicator vector $v \in \{0,1\}^w$

Transition can be expressed as matrix product corresponding to transition

width of branching program (captures "space" usage of program)

Example:



$M^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ transition for reading 0

$M^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ transition for reading 1

layered branching program: graph can be decomposed into layers, edges only between adjacent layers

on each layer, program reads 1 bit of the input (same bit of input is read for all nodes in the layer)

↳ Important: same bit of input can be read multiple times

Theorem (Barrington). Let $C: \{0,1\}^k \to \{0,1\}$ be a Boolean circuit with depth d and fan-in 2 (i.e., each gate has two inputs). Then, we can compute C using a permutation branching program of length $\ell \le 4^d$ and width 5.

transition matrix can be described by a permutation matrix

In particular, if $d = O(\log n)$, the length of the branching program is $\ell \le 4^d = 4^{O(\log n)} = \mathrm{poly}(n)$.

Let $BP = (\mathrm{inp}, M_{i,0}, M_{i,1})$ be a branching program on input $x \in \{0,1\}^n$ with length ℓ and width w:

- $\mathrm{inp}: [\ell] \to [n]$ specifies which bit of input to read in given layer
- $M_{i,0}, M_{i,1} \in \{0,1\}^{w \times w}$ specifies transition for reading 0 or 1 in layer i
- Let $v_0 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ be initial state.
- Let $t \in \{0,1\}^w$ be indicator for accepting states in output layer

- Can compute $BP(x)$ as:

$$BP(x) = t^T \cdot A_{\ell, x_{\mathrm{inp}(\ell)}} \cdot A_{\ell-1, x_{\mathrm{inp}(\ell-1)}} \cdots A_{1, x_{\mathrm{inp}(1)}} \cdot v_0$$

To compute homomorphically: given fresh encryptions of bits of x, homomorphically compute

$$A_{i, x_{\mathrm{inp}(i)}} = x_i \cdot A_{i,1} + (1 - x_i) \cdot A_{i,0}$$

← if encryptions of x have noise at most B, then encryptions of $A_{i,\mathrm{inp}(i)}$ has noise at most $2B$

Homomorphically compute sequence of product

$$t^T \cdot \boxed{A_{\ell, x_{\mathrm{inp}(\ell)}} \cdot A_{\ell-1, x_{\mathrm{inp}(\ell-1)}} \cdots A_{1, x_{\mathrm{inp}(1)}}} \cdot v$$

Observe: Each product involves at least one "fresh" ciphertext $A_{i, x_{\mathrm{inp}(i)}}$, so by asymmetric noise growth of GSW multiplication, overall noise is $\ell \cdot B \cdot \mathrm{poly}(m)$

Decryption circuit has depth $O(\log n)$ so associated branching program BP has length $4^d = \mathrm{poly}(n)$.

↳ Overall noise from bootstrapping: $\ell \cdot B \cdot \mathrm{poly}(m) = \mathrm{poly}(n)$

For correctness, it now suffices to use $q = \mathrm{poly}(n)$, so can get FHE with polynomial modulus q ← further improvements possible!