

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

# CSE208: Advanced Cryptography (FHE)

Daniele Micciancio

UCSD

Winter 2023



# Section 1

FHE!!

# Bootstrapping

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Given (1-hop)  $(\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$  supporting functions

$$f_{c,c'}(\text{sk}) = \text{Dec}(\text{sk}, c) \text{ nand } \text{Dec}(\text{sk}, c')$$

- Define (multi-hop) FHE scheme with  $\text{Func} = \{ \text{nand} \}$

```
Gen'() = (sk, pk) ← Gen()  
ek ← Enc(pk, sk)  
return (sk, (pk, ek))
```

```
Enc'((pk, ek), m) = Enc(pk, m)
```

```
Eval'((pk, ek), nand, c, c')  
= EvalC(pk, f_{c,c'}, ek)
```

# LWE Homomorphic Encryption

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Goal: homomorphic evaluation of

$$f_{c,c'}(sk) = \text{Dec}(sk, c) \text{ nand } \text{Dec}(sk, c')$$

- LWE-based cryptosystem
  - Supports bounded depth addition and multiplication
  - Bit operations:  $x \text{ nand } y = 1 - (1-x) \cdot (1-y)$

- Key Switching

$$ek[i] = \text{Enc}'(sk', sk[i])$$

$$\text{KeySwitch}(ek, (a[], b)) =$$

$$\text{Const}(b) - \sum_i \text{CMul}(a[i], ek[i])$$

- Homomorphic evaluation of  $\text{Dec}'(a, b) = b - Sa$

# Not enough

- Key switching only computes the linear part of Dec
- We also need to round the result to decode  $(b - Sa)$
- Is this really needed?
  - Yes,  $b - Sa = (q/p)m + e$
  - Key switching gives a noisy encryption of  $(q/p)+e$
  - Without rounding, noise keeps getting bigger

## Questions

- Can we express rounding as a polynomial function (mod  $q$ )?
- What is the degree of the polynomial?

# Error growth and bounded computation

We have seen two methods to multiply ciphertexts:

- Tensor products
  - error growth  $\sim \beta \rightarrow \beta\sigma$
  - can evaluate arbitrary circuits with **multiplicative depth**  $L$
  - even for  $L = \log n$ , requires superpolynomial modulus  $q > \sigma^L \approx n^{O(\log n)}$
- Nested Encryption / Homomorphic Decryption
  - asymmetric error growth:  $(m_0, e_0) \times (m_1, e_1) \rightarrow m_0 e_1 + e_0 \beta$
  - can evaluate arbitrary multiplication **chains** of  $L$  **fresh** encryptions of **binary** messages
  - even for large  $L$ , polynomial modulus  $q \approx L\beta^2$  is enough

# Roadmap

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

For each multiplication method

- 1 Describe/analyze a bootstrapping algorithm
- 2 Homomorphically evaluate the algorithm using an appropriate cryptographic data structure (encrypted accumulator)
- 3 Implement the cryptographic data structure using LWE

# Cryptographic accumulators

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Cryptographic Data Structure  $ACC[v]$ 
  - Holds a value  $v \in V$  in encrypted form
  - Input Encryption scheme:  $Enc'$
  - Output Encryption scheme:  $Enc''$
- Operations on  $ACC[v]$ 
  - Given  $Enc'(x)$ , update  $ACC[v] \rightarrow ACC[f(v, x)]$
  - Given  $ACC[v]$ , output  $Enc''(f(v))$
- Bootstrapping:
  - Bootstrapping key:  $Enc'(s)$
  - Final output:  $Enc''(m)$



# Boostrapping problem

- Assume  $p = 2, m \in \{0, 1\}$
- Decryption Algorithm:
  - Input:  $a[1..n] \in \mathbb{Z}_q^n, b \in \mathbb{Z}_q$
  - Secret key:  $s[1..n] \in \mathbb{Z}^n$
  - Compute  $d = b - \sum_i a[i]s[i] + (q/4) \pmod{q}$
  - Round  $d$  to  $MSB(d) = \lfloor 2d/q \rfloor$
- Homomorphic Computation:
  - Given  $Enc(s[i])$
  - Compute  $Enc(MSB(d))$
- Simplifying assumption:
  - $s[i] \in \{0, 1\}$
  - without loss of generality using  $(a, 2a, 4a, \dots)$

# Ripple-carry addition

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Standard schoolbook method
  - using binary digits
  - add  $n$  numbers at a time
  - *carry* in  $\{0, \dots, n\}$
- Input digits are encrypted

# Ripple-carry accumulator

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Parameters:
- Message space  $V = \{v', \dots, v''\}$
- Input:  $\text{Enc}'(x) = \text{Enc}^\#(x)$
- Output:  $\text{Enc}''(x) = \text{LWE}(x)$
- $\text{ACC}[x] = (\text{Enc}''("x=v")) : v \in V$ 
  - $\text{Init}(v) = \text{ACC}[v]$
  - Function application:  $f(\text{ACC}[v]) = \text{ACC}[f(v)]$
  - Selection:  
 $\text{Enc}'(b) ? \text{ACC}[v_0] : \text{ACC}[v_1] = \text{ACC}[b?v_0:v_1]$
  - Output:  $p(\text{ACC}[v]) = \text{Enc}''(p(b))$

# Bootstrapping algorithm

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

```
b+q/4 =  $\sum_j 2^j b[j]$ 
a[i] =  $\sum_j 2^j a[i,j]$ 
ACC  $\leftarrow$  ACC[0]
for h = 0..k-1
  ACC[x]  $\leftarrow$  f(ACC[x]) where  $f(x) = (x/2) + b[h]$ 
  forall i,j
    if (a[i,j] = 1)
      ACC[x + s[i]]  $\leftarrow$  Enc'(s[i]) ? ACC[x] :
        ACC[x+1]
return (even(ACC[x])) = Enc''(even(x))
```

# Carry-save accumulator

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Parameters: bit length  $k$
- $\text{ACC}[x] = (x_0, x_1)$ 
  - $x = x_0 + x_1$  (modulo  $2^k$ )
  - $x_0[0, \dots, k-1]$  and  $x_1[0, \dots, k-1]$
  - redundant representation
- Operations:
  - add  $y$  to ACC
  - compute  $\text{MSB}(\text{ACC})$

# Carry-save addition

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

```
Add(ACC(x0, x1), y):
```

```
  x0'[i] = (x0[i] + x1[i] + y[i]) mod 2
```

```
  x1'[i+1] = (x0[i] + x1[i] + y[i] > 1)
```

```
  return ACC(x0', x1')
```

FHE!!

# MSB computation

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Standard MSB computation
  - addition  $x_0+x_1$  with carry propagation
  - $O(\log(k))$  depth circuit where  $k=\log(q)$
- Can also add in  $\log(k)$  depth
  - Compute both  $\text{MSB}(\text{ACC})$  and  $\text{MSB}(\text{ACC}+1)$
  - $\text{ACC}[k]$ :  $k$ -bit accumulator
  - Recursive algorithm: split  
 $\text{ACC}[k] = (\text{HiACC}[k/2], \text{LoACC}[k/2])$

`MSBs(ACC=(HiACC, LoACC)):`

`parallel:`

`hi[0,1] = MSBs(HiACC)`

`lo[0,1] = MSBs(LoACC)`

`out[0] = hi[lo[0]]`

`out[1] = hi[lo[1]]`

`return out`

# Bootstrapping algorithm

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

```
ACC[0] = b+q/4
for i=1..n
    ACC[i] = s[i]*a[i]
ACC = Sum(ACC[0],...,ACC[n])
return MSB(ACC)
```

```
Sum(ACC[0..n])
    if n=0
        then return ACC[0]
    else h=n/2
        ACC0 = Sum(ACC[0..h-1])
        ACC1 = Sum(ACC[h..n])
        (x0, x1) = ACC1
        return (ACC0 + x0) + x1
```



# Summary

Bootstrapping functions can be computed by

- 1  $O(n \log q)$ -long sequence of multiplications, or
- 2  $\log(n) + \log \log(q)$ -depth arithmetic circuits

Error growth:

- 1 Using **LWE**  $\odot$ : final error  $\approx O(n) \cdot \beta$
- 2 Using **LWE**  $\otimes$ : final error  $\approx \sigma^{\log n + \log \log q} = \sigma^{O(\log n)}$

Parameters  $\beta(n), \sigma(n)$ : fixed polynomials in  $n$

Modulus:

- 1 polynomial modulus  $q(n) \approx O(n)\beta = n^{O(1)}$
- 2 quasipolynomial  $q(n) = n^{O(\log n)}$

# Summary (security)

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Hardness of lattice problems within factor  $\gamma \approx q/\beta$ 
  - 1 LWE  $\odot$ : polynomial  $\gamma = n^{O(1)}$
  - 2 LWE  $\otimes$ : quasipolynomial  $\gamma = n^{O(\log n)}$
- Circular security assumption
  - Needed by tensor product multiplication / keyswitching
  - Needed to apply bootstrapping
  - Not needed for leveled homomorphic encryption

# Summary (security)

- Hardness of lattice problems within factor  $\gamma \approx q/\beta$ 
  - 1 **LWE**  $\odot$ : polynomial  $\gamma = n^{O(1)}$
  - 2 **LWE**  $\otimes$ : quasipolynomial  $\gamma = n^{O(\log n)}$
- Circular security assumption
  - Needed by tensor product multiplication / keyswitching
  - Needed to apply bootstrapping
  - Not needed for leveled homomorphic encryption

## Question

*Remove circular security assumption:*

- *Can you build (unbounded) FHE from standard LWE?*
- *Can you build (unbounded) linearly homomorphic HE?*

# Efficiency

CSE208:  
Advanced  
Cryptography  
(FHE)

Daniele  
Micciancio

FHE!!

- Main security parameter  $n > 100$  (typically,  $n \approx 1000$ )
- Modulus  $q(n) < 2^n$  has bitsize  $\log q < n$
- Assume 1GHz, arithmetic operations modulo  $q$
- Bootstrapping: homomorphically evaluate decryption algorithm (once or twice per gate)

## Question

*Can you estimate the cost of a single FHE operation?*