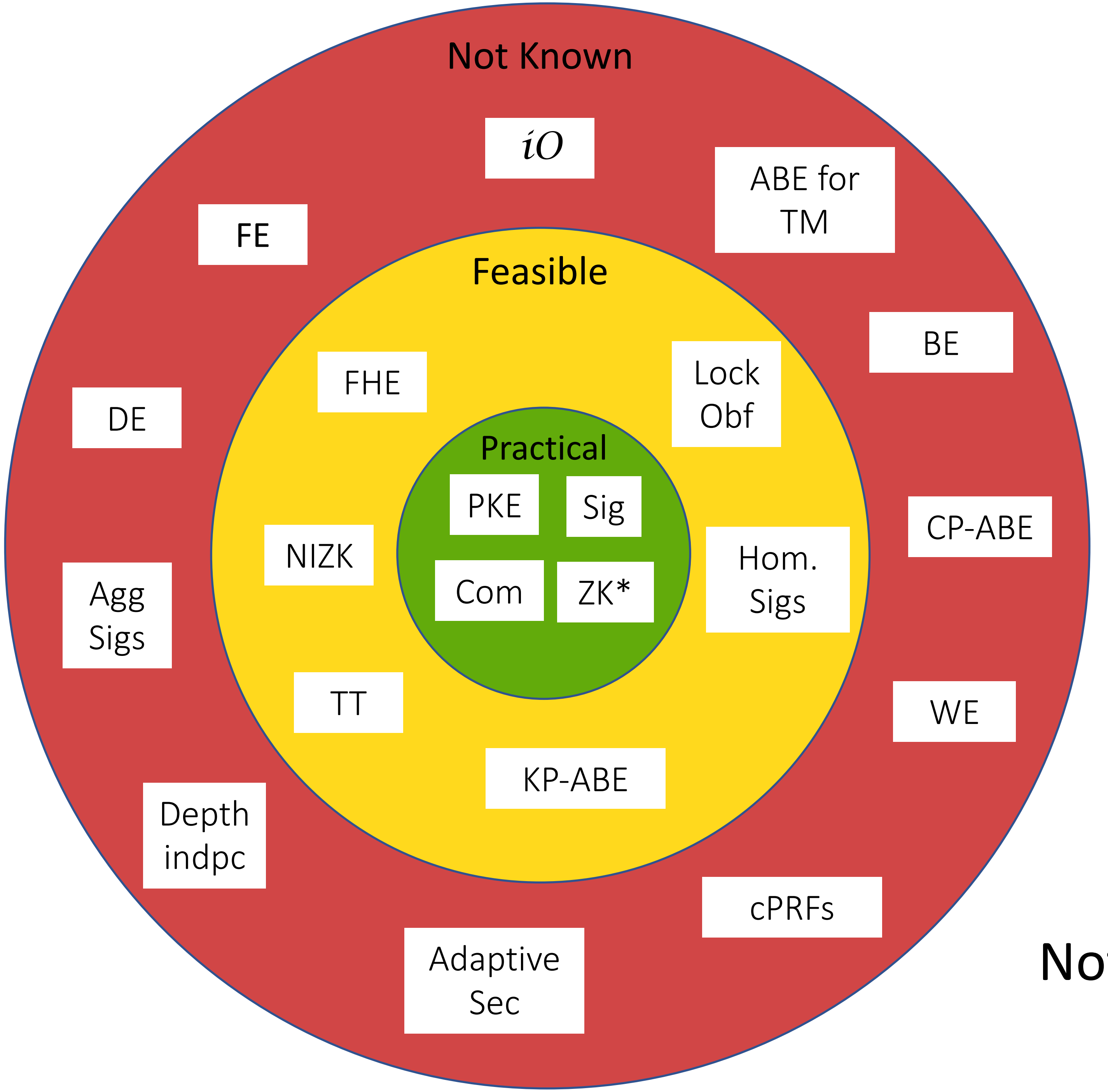
Recent Progress & Challenges in Lattice Based Cryptography

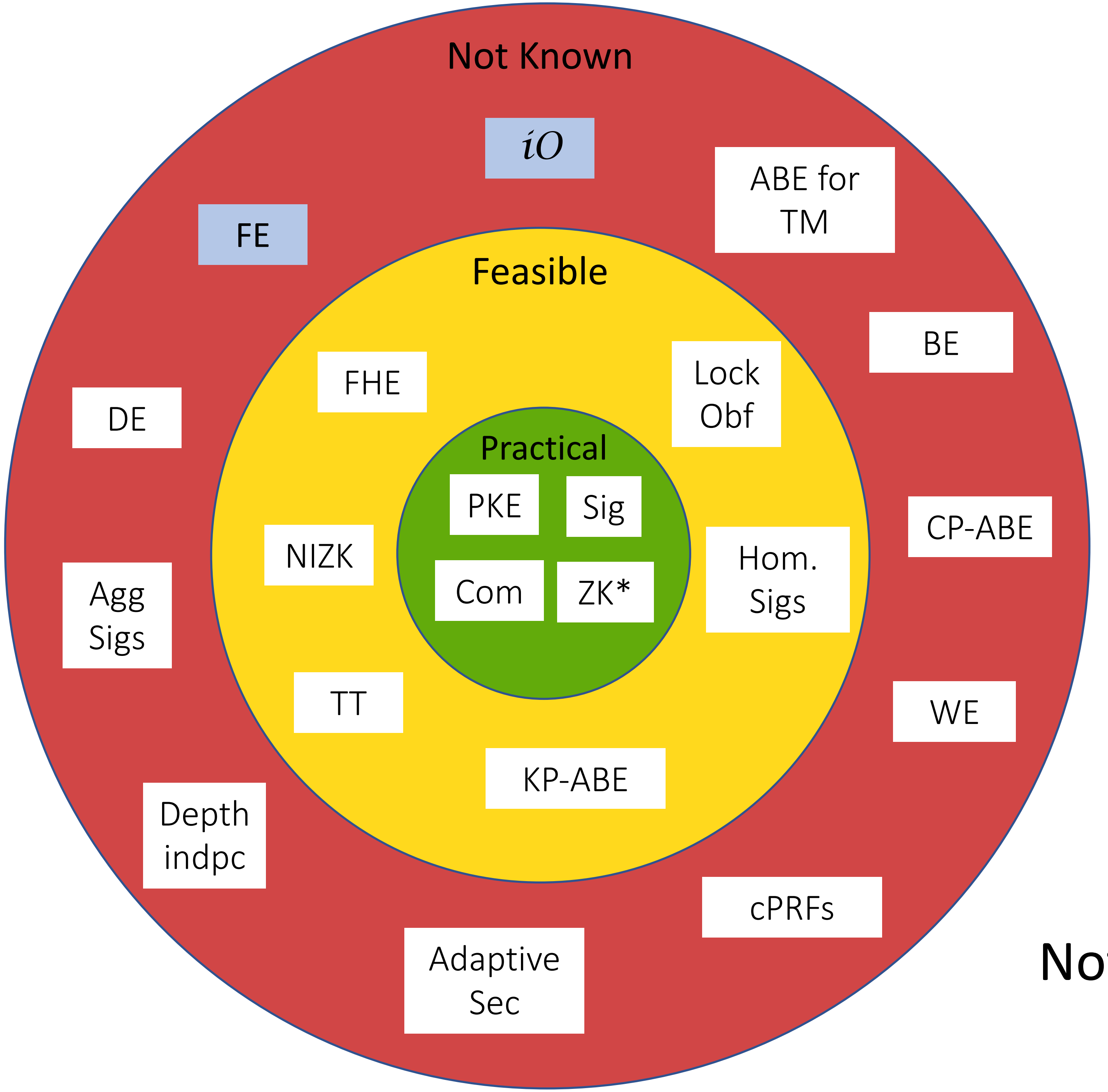
Shweta Agrawal
IIT Madras

LATTICE CRYPTO



Not Exhaustive!

LATTICE CRYPTO



Not Exhaustive!

Road Map

- Obfuscation
- Functional Encryption
- Overview of Progress
- Challenges from Lattices
- Open Problems

Obfuscation

I know a better algorithm to factor numbers!

Win best paper award at Crypto?

Keep algorithm secret and sell functionality.

code → OBFUSCATOR →

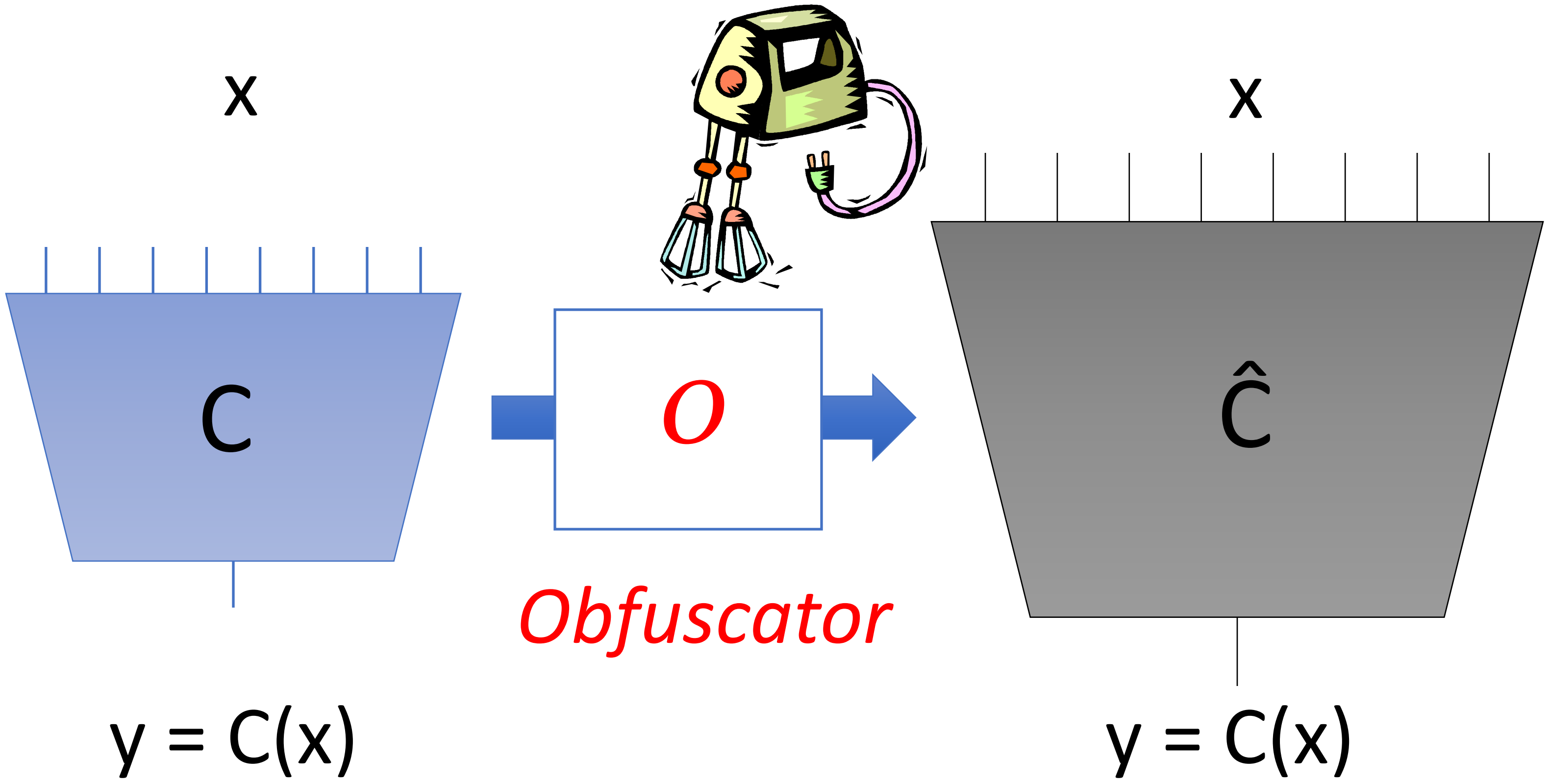
Obfuscated code

```
#include<stdio.h> #include<string.h>
main(){char*O,l[999]="'`acgo\177~|xp .
-\OR^8)NJ6%K4O+A2M(*0ID57$3G1FBL";
while(O=fgets(l+45,954,stdin)){*l=O[
strlen(O)[O-1]=0,strspn(O,l+11)];
while(*O)switch((*l&&isalnum(*O))-!*l)
{case-1:{char*I=(O+=strspn(O,l+12)
+1)-2,O=34;while(*I&3&&(O=(O-16<<1)+
*I---'-')<80);putchar(O&93?*I
&8||!( I=memchr( l , O , 44 ) ) ?'?':
I-l+47:32); break; case 1: ;}*l=
(*O&31)[l-15+(*O>61)*32];while(putchar
(45+*l%2),(*l=*l+32>>1)>35); case 0:
putchar((++O ,32));}putchar(10);}}
```

- Produces correct output
- Impossible to reverse engineer

Obfuscation

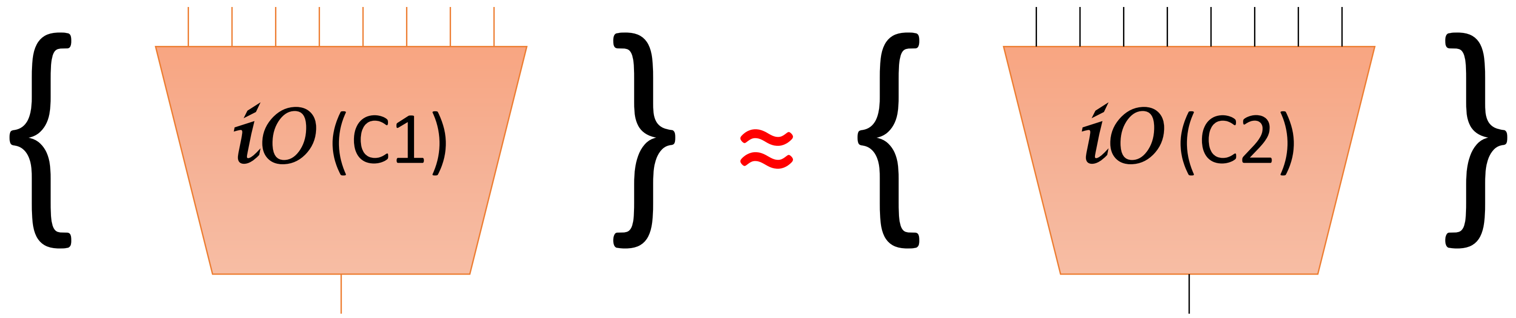
Compile a circuit C into one Ĉ that *preserves functionality,*
and is unintelligible (resistant to reverse engineering)



Indistinguishability Obfuscator $i\mathcal{O}$ [BGI+01]

"Which one of two equivalent circuits $C_1 \equiv C_2$ is obfuscated?"

$C_1 \equiv C_2$, meaning
- Same size $|C_1|=|C_2|$
- Same truth table $TB(C_1) = TB(C_2)$



Trivial if efficiency is not a concern

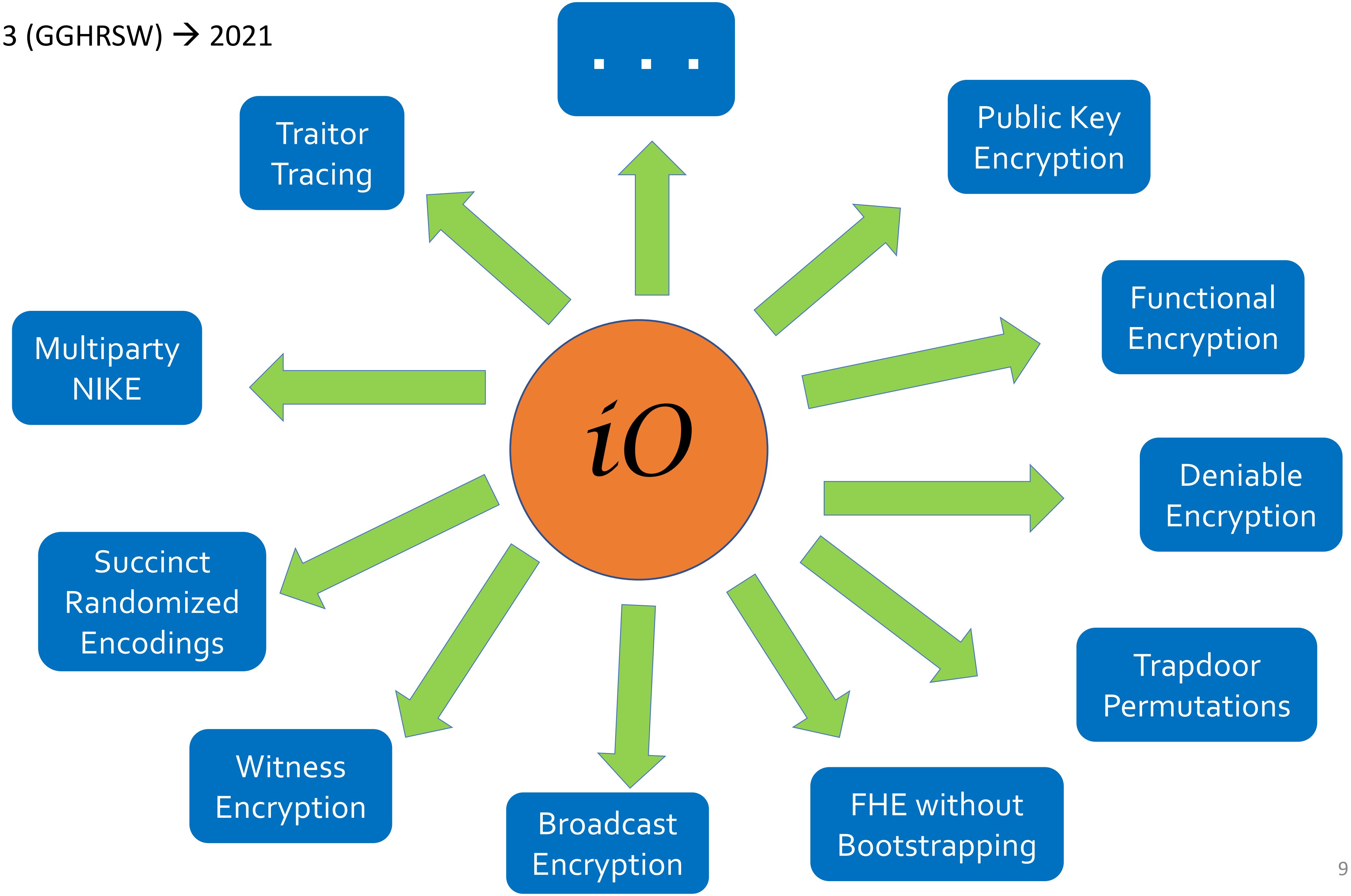
Goal: Find an efficient compiler $i\mathcal{O}$

Before we proceed… why do we care?

- Seemingly useless definition
- We already know both circuits are equivalent. Does it matter what is the particular representation?
- Unclear if there are applications

"Theorem" (GGHRSW13,SW13…) : $i\mathcal{O}$ is (almost?) crypto-complete

2013 (GGHRSW) → 2021



Constructing $i\mathcal{O}$: Broadly Two Approaches

- Direct Constructions
  - All based on "multilinear maps" [GGH13,CLT13,GGH15]
  - Constructed from lattices
  - Many attacks, fixes, repeat: hard to understand security

- Bootstrapping based constructions

Recap: Bilinear Maps

- Cryptographic bilinear map
  - Groups G_1 and G_2 of order p with generators g_1, g_2 and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ such that

$$\forall\, a, b \in Z_p^*, \qquad e\left(g_1^a, g_1^b\right) = g_2^{ab}$$

- Hardness (Bilinear Diffie Hellman): Can compute degree 2 “in the exponent”, degree 3 looks like random.

- Efficient Instantiation: Weil or Tate pairings over elliptic curves.

- Tremendously useful for crypto!

Multilinear Maps: Classical Notion

- Cryptographic n-multilinear map (for groups)
  - Groups $G_1, \ldots, G_n$ of order p with generators $g_1, \ldots, g_n$
  - Family of maps:

    $e_{i,k}\colon G_i \times G_k \to G_{i+k}$ for $i + k \leq n$, where

    - $e_{i,k}(g_i^a, g_k^b) = g_{i+k}^{ab}\ \forall a, b \in Z_p$.

  - Hardness: at least "discrete log" in each G_i is "hard".
    - And hopefully the generalization of Bilinear DH

Multilinear Maps

- Applications described by Boneh and Silverberg in 2003
  - Pessimistic about existence in realm of algebraic geometry
- First candidate construction by Garg, Gentry, Halevi, 2013
  - Based on ideal lattices, ideas inspired by NTRU
- Immensely useful, can be used to build $i\mathcal{O}$ (and much more!).

Where are we with this?

Multilinear Maps

We still **don't have candidates** of "clean multi-linear" maps

Noisy multilinear maps:

[Garg-Gentry-Halevi13, Garg-Gentry-Halevi-Raykova-Sahai-Waters13, Coron-Lepoint-Tibouchi13, Gentry-Gurbonov-Halevi15 , Coron-Lepoint-Tibouchi15,…]

Linearization attacks

All broken!

: [Miles-Sahai-Zhandry16, Apon-DGargM17, Cheon-Han-Lee-Ryu-Stehle15, Coron-Gentry-Halevi-Lepoint-Maji-Miles-Raykova-Sahai-Tibouchi15]

Slide Credit: Yael Kalai

Multilinear Maps

[Badninarayanan-Miles-Sahai-Zhandry16, Garg-Miles-Mukherjee-Sahai-Srinivasan-Zhandry16]:

IO assuming Weak MMAPs

Not broken (yet…)

Noisy multilinear maps:

[Garg-Gentry-Halevi13, Garg-Gentry-Halevi-Raykova-Sahai-Waters13, Coron-Lepoint-Tibouchi13, Gentry-Gurbonov-Halevi15 , Coron-Lepoint-Tibouchi15,…]

Linearization attacks

All broken!

: [Miles-Sahai-Zhandry16, Apon-DGargM17, Cheon-Han-Lee-Ryu-Stehle15, Coron-Gentry-Halevi-Lepoint-Maji-Miles-Raykova-Sahai-Tibouchi15]

Slide Credit: Yael Kalai

Multilinear Maps

[Badninarayanan-Miles-Sahai-Zhandry16, Garg-Miles-Mukherjee-Sahai-Srinivasan-Zhandry16]:

IO assuming Weak MMAPs

Not broken (yet…)

Generation 1 iO (poly degree maps)

[GGHRSW13, BGKPS14, BR14, PST14, AGIS14, BMSZ16, CLT13, CLT15, GGH15, MSZ16, GMMSSZ16]

Open #1: Improve security from lattices

Bootsrapping Based Constructions: Reduce, Reduce, Reduce

- What is the minimum functionality needed for iO?
- How much can we “clean up” assumptions?
- Sequence of works reduced degree from poly to constant

Generation 2 iO (constant degree maps)

Lin16, LV16, AS17, LT17

Bootsrapping Based Constructions: Reduce, Reduce, Reduce

Generation 3 iO (maps <= 2, small heuristic component)

AJLMS 19, A19, JLMS19, JLS19, AP20, GJLS20

Generation 4 iO (maps <= 2 + additional assumptions)

JLS21, GP21, BDGM21, WW21, DQVWW21

JLS21: standard assumptions (SXDH, LWE, LPN, PRG in NC0)
Others: some nonstandard-ness (eg LWE w/ circularity)

Bootsrapping Based Constructions: Reduce, Reduce, Reduce

Generation 4 iO (maps <= 2 + additional assumptions)

JLS21, GP21,BDGM21,WW21,DQVWW21

JLS21: standard assumptions (SXDH, LWE, LPN, PRG in NC0)
Others: some nonstandard-ness (eg LWE w/ circularity)

Open #2: Post quantum iO from standard assumptions

Post quantum iO (with provable security)
What approaches do we have?

- Functional Encryption
- Functional Encodings (or succinct randomized encodings) (WW21, DQVWW21)
- Circularity assumptions on FHE (BDGM21, GP21)

Most open, will focus on this

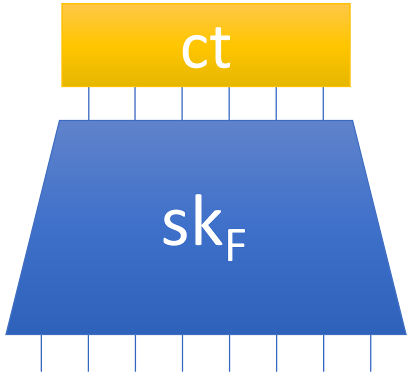
Functional Encryption
Encryption with Partial Decryption Keys

(mpk, msk) ← Setup(1^n)

Encrypt (mpk, x):

Keygen(msk, F):

Decrypt (sk_F, ct):



Security:
Adversary possessing keys for multiple circuits F_i cannot distinguish $Enc(x_0)$ from $Enc(x_1)$ unless $F_i(x_0) \neq F_i(x_1)$

Functional Encryption [SW05,BSW11]

FE → iO [AJ15, BV15,Lin16,LV16,AS16]

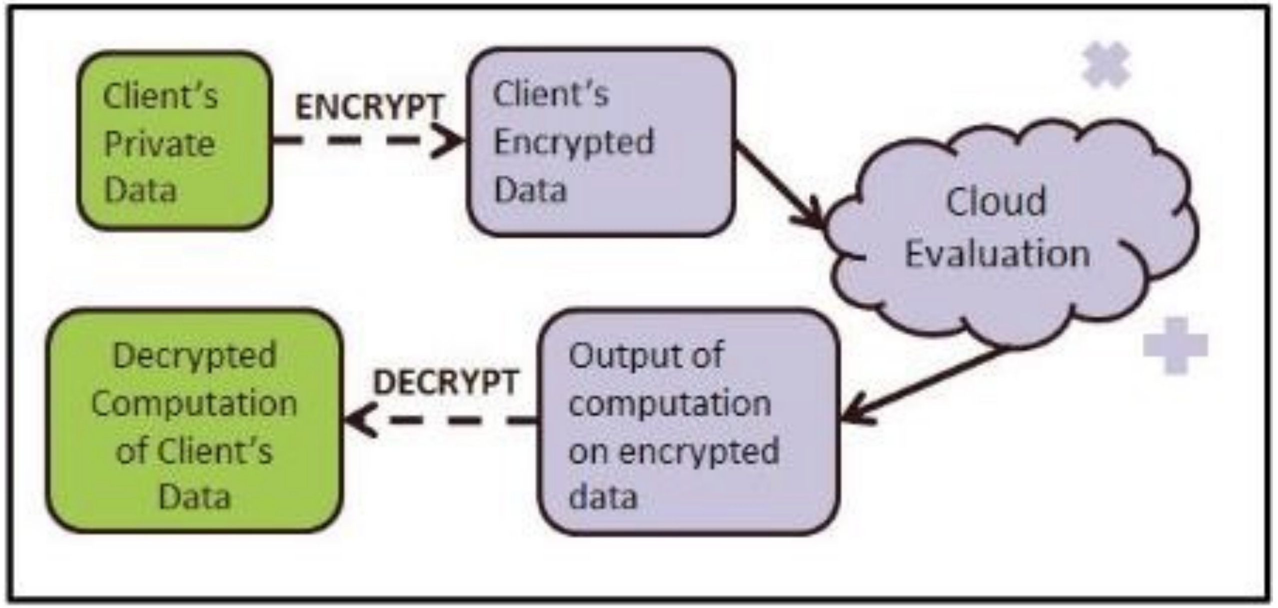
The following FE suffices for iO:

- Single key for a function with long output $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$
- $|CT|$ is sublinear in output length m
- Supporting function class NC^0

How to build it?

Natural Idea: Use LWE

-- Recall: LWE *only* assumption yielding FHE



Expressive Functionality: Supports arbitrary circuits

Compact ciphertext, independent of circuit size

Perfect: Encrypted computation with All or Nothing Decryption

* : up to minor variations

LWE → Leakage on Partial Decryption

- Using LWE, can support all polynomial sized circuits for FE
- But only for restricted security games
  - Adversary sees limited number of queries [GVW12, GKPVZ13, AR17], restricted types of queries [GVW15], combination of these [A17]
- Attacks against scheme when adversary violates security game [A17]

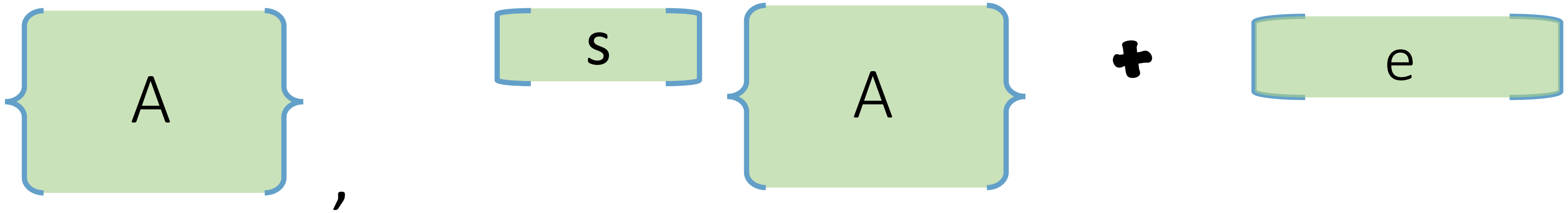
Causes of Attack and Ways to Overcome them?

Challenge: Leaky LWE Keys

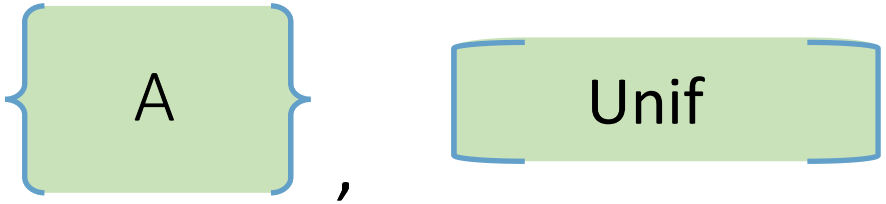
In Most LWE Based FE Constructions

Learning With Errors → Ciphertext

Distinguish “noisy inner products” from uniform



versus

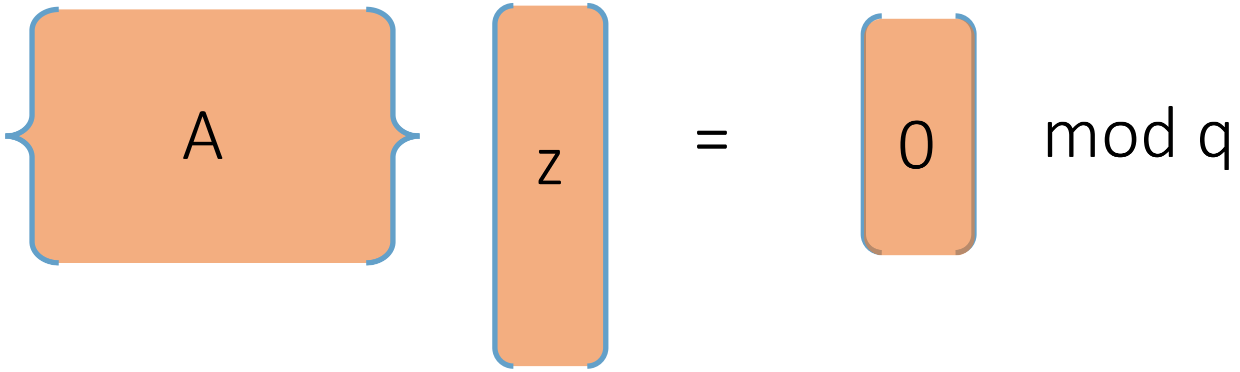


In Most LWE Based FE Constructions

SIS Problem ➔ Secret Key

Given matrix A, find "short" integer z such that

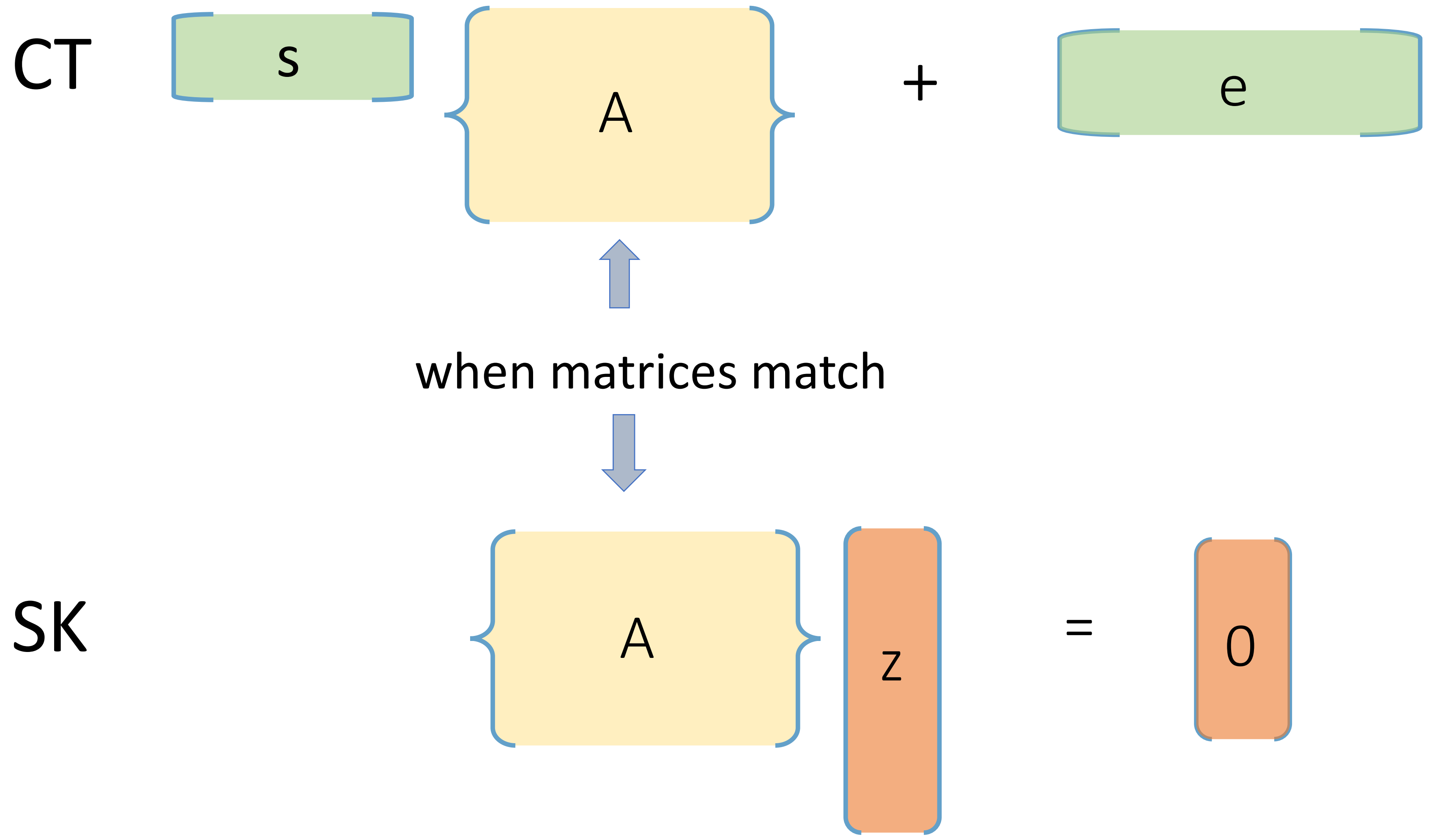
$A z = 0 \bmod q$



Many short vectors form a trapdoor for A

Can be used to break LWE with matrix A

Decryption works



We need: Partial Decryption Capability

Encrypt (mpk, x):

LWE encodings of x

c_1 = A_1, x_1 …….. c_n = A_n, x_n

c_0 = A, 0

We need: Partial Decryption Capability

BGG+14 showed homomorphic evaluation algorithms $eval_{ct}$ and $eval_{pk}$ such that:

Encrypt (mpk, x):

LWE encodings of x

c_1 = A_1, x_1 .... c_n = A_n, x_n

c_0 = $A, 0$

1. Compute $A_f = Eval_{pk}(A_1 \ldots A_n, f)$

1. Compute $ct^* = Eval_{ct}(c_1 \ldots c_n, f)$

$ct^* = [A | A_f], f(x)$

We need: Partial Decryption Capability

BGG+14 showed homomorphic evaluation algorithms $eval_{ct}$ and $eval_{pk}$ such that:

Encrypt (mpk, x):

LWE encodings of x

c_1 = [A_1, x_1] …….. c_n = [A_n, x_n]

c_0 = [A, 0]

Keygen(msk, f):

1. Compute $A_f = Eval_{pk}(A_1 \ldots A_n, f)$
2. Compute short vector z such that

$$\{A \mid A_f\} \, [z] = [0]$$

Decrypt (sk_f, ct) ➔ f(x)

1. Compute $ct^* = Eval_{ct}(c_1 \ldots c_n, f)$

ct^* = [$[A \mid A_f]$, f(x)]

We need: Partial Decryption Capability

BGG+14 showed homomorphic evaluation algorithms $eval_{ct}$ and $eval_{pk}$ such that:

Encrypt (mpk, x):

LWE encodings of x

c_1 = A_1, x_1 …….. c_n = A_n, x_n

c_0 = A, 0

Keygen(msk, f):

1. Compute $A_f = Eval_{pk}(A_1 \ldots A_n, f)$
2. Compute short vector z such that

$$[A \mid A_f] \, z = 0$$

Decrypt (sk_f, ct) → f(x)

1. Compute $ct^* = Eval_{ct}(c_1 \ldots c_n, f)$

$ct^* = [A|A_f] , f(x)$

Matrices in ct* and key match, can recover f(x) !

Catch: x is not hidden

We need: Partial Decryption Capability

GVW15 showed how to hide x in restricted security game

Encrypt (mpk, x): Use FHE to encrypt x_i
denote by $\hat{x}_i$

c_1 = $A_1, \hat{x}_1$ …….. c_n = $A_n, \hat{x}_n$

c_0 = A, 0 c_{sk} = FHE.sk

Keygen(msk, f): Let f' = FHE.Dec ∘ f

We need: Partial Decryption Capability

GVW15 showed how to hide x in restricted security game

Encrypt (mpk, x): Use FHE to encrypt x_i
denote by $\hat{x}_i$

$c_1 = \boxed{A_1, \hat{x}_1}$ .... $c_n = \boxed{A_n, \hat{x}_n}$

$c_0 = \boxed{A, 0}$ $c_{sk} = \boxed{\text{FHE.sk}}$

Keygen(msk, f): Let f' = FHE.Dec ∘ f

1. Compute $A_{f'} = \text{Eval}_{PK}(A_1 \ldots A_n, f')$
2. Compute short vector z such that

$$\left[A \,|\, A_{f'} \right] z = 0$$

Decrypt (sk_f, ct) → f(x)

1. Compute $ct^* = \text{Eval}_{ct}(c_1 \ldots c_n, f')$

$ct^* = \boxed{[A|A_{f'}], f(x)}$

OK to reveal $\hat{x}_i$
Need work to argue that FHE.sk is hidden

Can be done in <u>restricted security game</u>, where Adv may not request any keys such that f(x) =1

Attacks Outside Game[A17]

- Request keys for linearly dependent vectors
- Combine keys to get short vectors, hence trapdoor in certain lattice A*
- Manipulate challenge CT to get LWE sample with matrix B*
- A* and B* only match for keys where f(x)=1
- Lessons: *Inherent vulnerability* for “attribute hiding” scheme with this structure of keys

How do pairings help [GJLS20]?

Can build FE for quadratic functions from pairings [Lin16,BCFG17,G20,Wee20]

(mpk, msk) ← Setup(1^n)

Encrypt (mpk, x = (x_1….x_n)):

ct

Keygen(msk, C = (c_{11}….c_{nn})):

Decrypt (sk_C, ct) outputs

$$\sum_{i,j} c_{ij}\, x_i x_j$$

No restrictions in the security game!

How do pairings help [GJLS20]?

- Compute ct* = [A|A$_{f'}$] , f(x) as before using evaluation algorithm
- Looking more closely at structure of ct*:

$$ct^* = [A|A_f]^T s + f(x) + noise$$

- Encryptor knows (s, noise) and can provide Linear FE ciphertext for vector (s, noise)
- Key generator knows [A| A_f] and can provide Linear FE key for vector ([A| A_f]^T, 1)
- Decryption recovers inner product ([A| A_f]^T s + noise, which can be subtracted from ct* to recover f(x) (upto rounding).

Using Pairing based FE to implement Quadratic (hence Linear) FE prevents the leakage created by LWE secret keys

Doing Without Pairings?

- Linear FE exists from LWE [ABDP15, ALS16] but does not suffice : same key structure
- There are other approaches [A19,AP20], but all suffer from *unsimulatable* key structure –
  - No known attacks but do not admit proof

Challenge: Construct LWE based FE with more secure keys

Challenge: leakage via FHE noise

Say we have secure keys…

- Pairings let us have secure keys.. are we done?
- Recall, challenge CT

$$ct^* = [A|A_f]^T s + f(x) + noise$$

- Decryption lets us get f(x) + noise
- Noise leaks too much information about x

Idea (AR17,A19,AJLMS19): flood noise to wipe out leakage

How to add flooding noise?

- Problem: Noise is too long! FE will not be compact ☹
- Idea: Use PRG – use seed in encrypt, expand during decrypt
- Problem: Need PRG in degree 2 to use with pairings, but degree 2 PRG insecure – LV18,BBKK18,BHJKS19
- Can we flatten the degree of computation so *public computation is high degree and private computation low degree (deg <= 2)*?
  - Idea used in FE before – GVW12, GVW15, AR17 ….

Deep, public computation done publicly, shallow private computation, done using linear/quadratic Functional Encryption

Degree Flattening

Given: LWE encoding of input x (encoding may vary).

Want: to compute a "deep" (say NC_1) circuit f on x, to obtain an encoding of f(x)

Can represent deep computation f as equivalent function f' such that f' has public computation of high degree and private computation of low degree

Can build

- FE for quadratic functions from pairings [Lin16,BCFG17,G20,Wee20]
- FE for linear functions from LWE, DCR, DDH [ABDP15, ALS16]

Linear Functional Enc [ABDP15,ALS16]

(mpk, msk) ← Setup(1^n)

Encrypt (mpk, x = (x_1....x_n)):

ct

Keygen(msk, y = (y_1....y_n)):

Decrypt (sk_y, ct) outputs

$$\sum_{i \in [n]} x_i y_i$$

No restrictions in the security game

More than n key requests
➔ MSK leaked

Symmetric key FHE for Quadratic Polynomials [BV11a]

s: secret key

Encrypt (s, x_1, x_2):

Sample u_1, u_2 randomly in ring. Sample err_1, err_2.

Compute :

$$c_1 = u_1 s + err_1 + x_1$$

$$c_2 = u_2 s + err_2 + x_2$$

Evaluate (c_1, c_2, $f = x_1 x_2$):

Want: Use c_1, c_2 to compute product ciphertext c_{12} that encrypts $x_1 x_2$

FHE Evaluation

We may write:

$$x_1 \approx c_1 - u_1 s$$

$$x_2 \approx c_2 - u_2 s$$

$$\therefore x_1 x_2 \approx c_1 c_2 - (c_1 u_2 + c_2 u_1) s + u_1 u_2 s^2$$

Let $c^{mult} = (c_1\, c_2, \quad c_1\, u_2 + c_2\, u_1, \quad u_1 u_2)$

Decryption $x_1 x_2 \approx \; < (c_1 c_2,\ (c_1 u_2 + c_2 u_1),\ u_1 u_2)\ ;\ (1, -s, s^2) >$

Degree Flattening [AR17]

- Recall FHE decryption equation:

$$x_1 x_2 \approx c_1 c_2 - (c_1 u_2 + c_2 u_1) s + u_1 u_2 s^2$$

- What if we group the t … lifferently?

$$\therefore x_1 x_2 \approx c_1 c_2 - \ldots) + \ldots$$

Known to encryptor

Known to Key Generator

Decryption

$$x_1 x_2 \approx c_1 c_2 + \langle (c_1 s,\ c_2 s,\ s^2) ; (-u_2, -u_1, u_1 u_2) \rangle$$

Degree Flattening [AR17]

Encryptor provides $c_1, \ldots\ldots c_n$ as well as Linear FE encryption of vector

$$(c_1 s,\ c_2 s\ ,\ldots.c_n s\ ,s^2)$$

Key Generator provides Linear FE key for vector

$$(-u_2, -u_1, 0\ldots..0,\ u_1 u_2)$$

Deep Computation is on public encodings

Computing $c_1 c_2$ herself, decryptor can recover :

$$x_1 x_2 \approx c_1 c_2 - u_2(c_1 s) - u_1(c_2 s) + u_1 u_2 (s^2)$$

Key Dependent Computation is Linear

Key Insight: Quadratic terms are $c_i c_j$ which are public

Only 2n ciphertexts instead of n^2

Can be generalized to NC_1 [AR17]

Use to generate noise?

- Last slide: Degree reduction to linear (LWE/DDH…)
  - Adversary sees exact linear equations in secrets
  - Too much leakage!
- GJLMS19: Degree reduction to quadratic (pairings)
  - Adversary sees quadratic equations in secrets
  - May be secure (aka MQ assumption for some distribution)
  - “Weak LWE with Leakage”
- JLS21: Use LPN (!!) to resolve leakage

Open #3: Quadratic FE from LWE?

Summary: Three Nuggets for Thought

How to Strengthen LWE keys

How to generate smudging noise?

Can we leverage fact that noise can be only approximately correct?

Open Problems

- Replace pairings with some weaker structure that can be built from LWE?
- New, simpler, plausible assumptions from lattices? Chart territory between LWE and multilinear map assumptions?
- Improve lattice based multilinear maps or $i\mathcal{O}$?
- Build post quantum FE and base applications on this?
- Use LWE for applications of $i\mathcal{O}$? Eg. Deniable Encryption.

Thank You

Images Credit:
Hans Hoffman
Jackson Pollock