# Obfuscation from Noisy Linear FE

Shweta Agrawal

IIT Madras

# 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

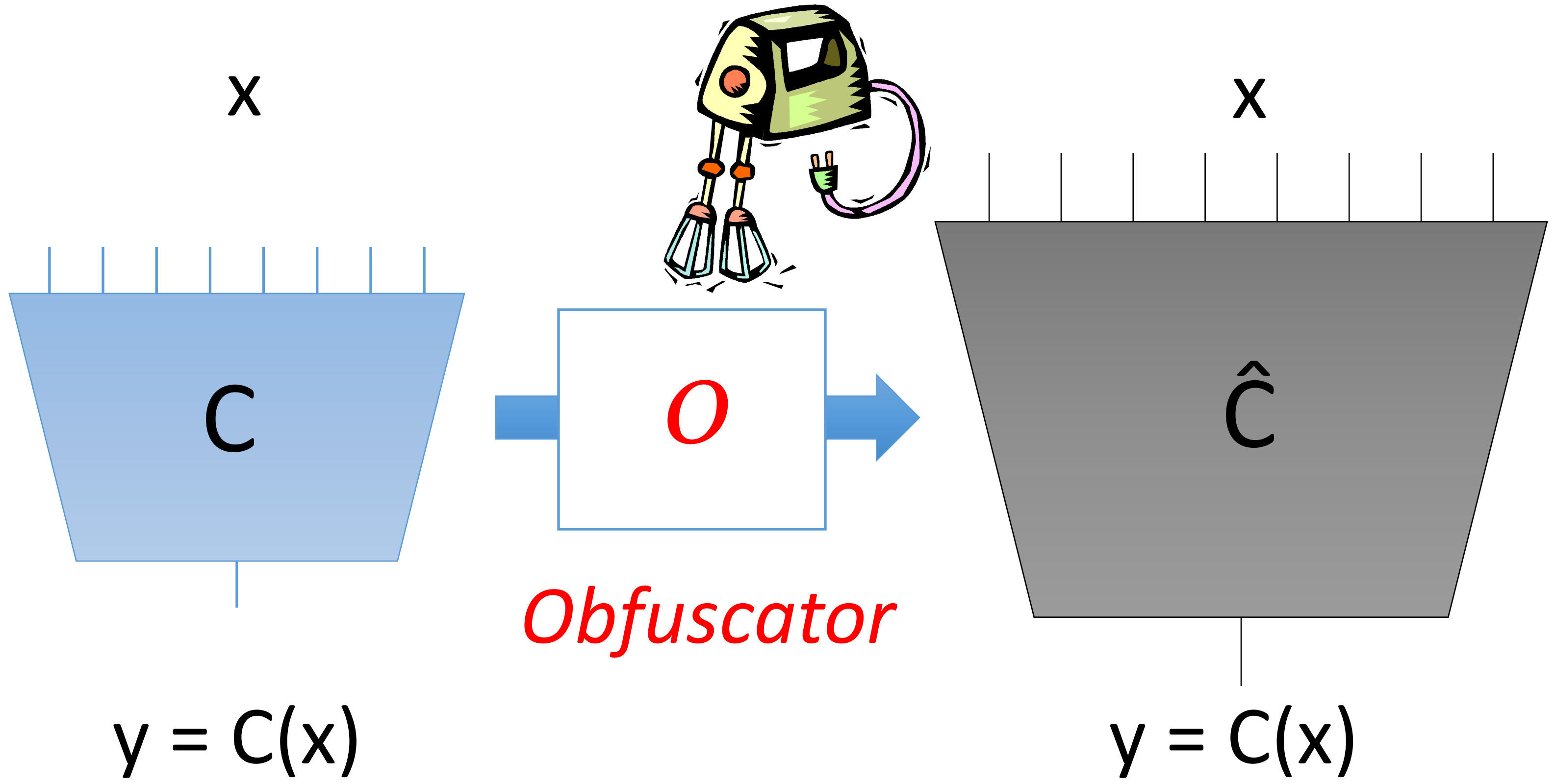
```
#include<stdio.h> #include<string.h>
main(){char*O,l[999]="'‘acgo\177~|xp .
-\OR^8)NJ6%K4O+A2M(*0ID57$3G1FBL";
while(O=fgets(l+45,954,stdin)){*l=O[
strlen(O)[O-1]=0,strspn(O,l+11)];
while(*O)switch((*l&&isalnum(*O))-!*l)
{case-1:{char*I=(O+=strspn(O,l+12)
+1)-2,O=34;while(*I&3&&(O=(O-16<<1)+
*I---’-’)<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,*

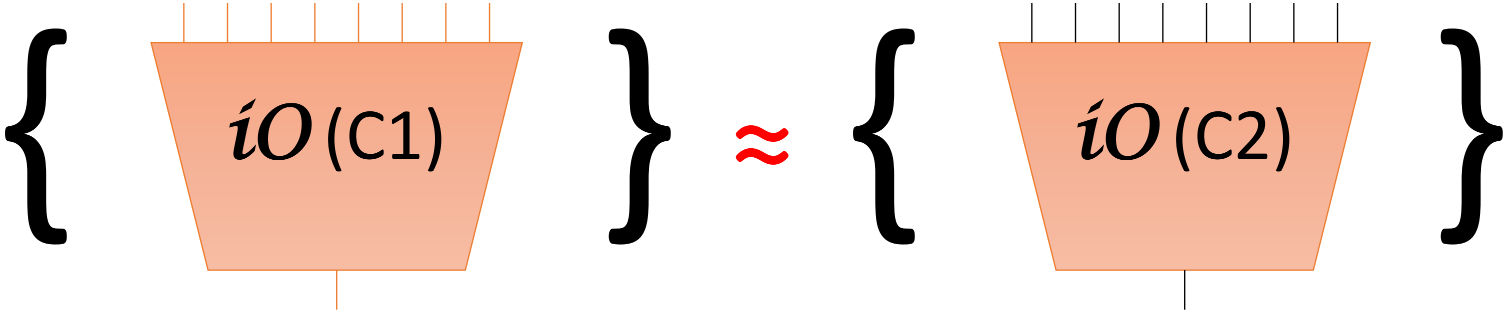
*and is unintelligible (resistant to reverse engineering)*



# 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

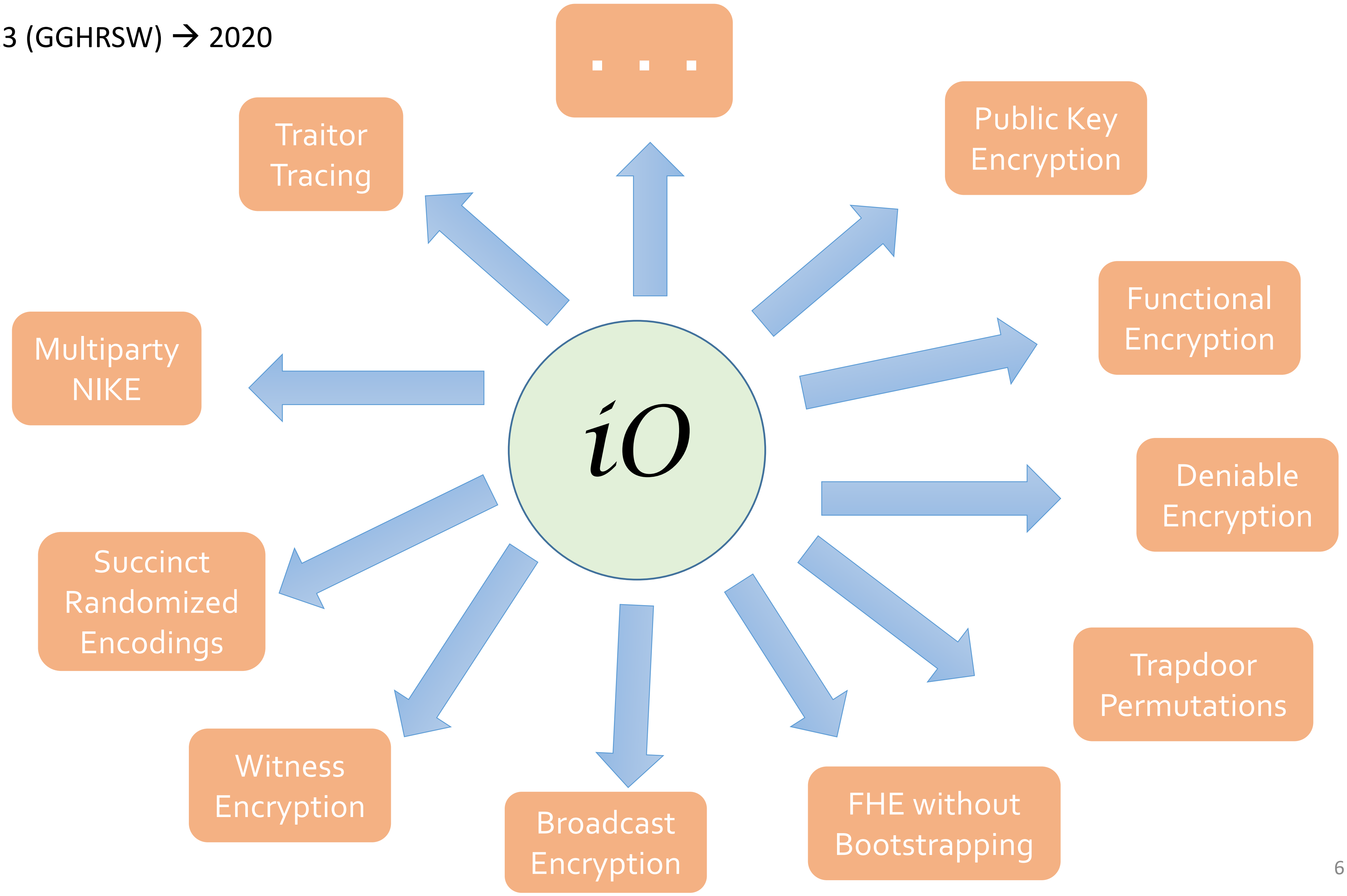
Goal: Find an efficient compiler $i\mathcal{O}$

# 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) → 2020



# Does $i\mathcal{O}$ exist?

# It Depends....

- Direct Constructions
  - All based on “multilinear maps” [GGH13,CLT13,GGH15]
  - Same template in all works (all eggs in same basket?)
  - Many attacks, fixes, repeat: hard to understand security

- Bootstrapping based constructions

# It Depends….

- Direct Constructions
  - All based on “multilinear maps” [GGH13,CLT13,GGH15]
  - Same template in all works (all eggs in same basket?)
  - 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 $G_1, \dots, G_n$ of order $p$ with generators $g_1, \dots, g_n$
  - Family of maps:

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

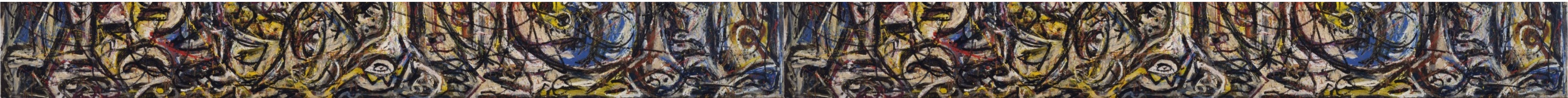
# Multilinear Maps

- Applications described by Boneh and Silverberg in 2003
  - Pessimistic about existence in realm of algebraic geometry

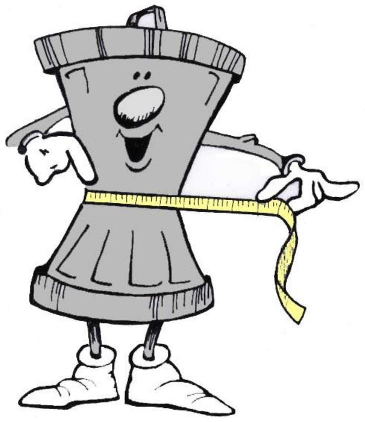
Yilei's talk

- First (beautiful!) candidate construction by Garg, Gentry, Halevi, 2013
  - Based on ideal lattices, ideas inspired by NTRU

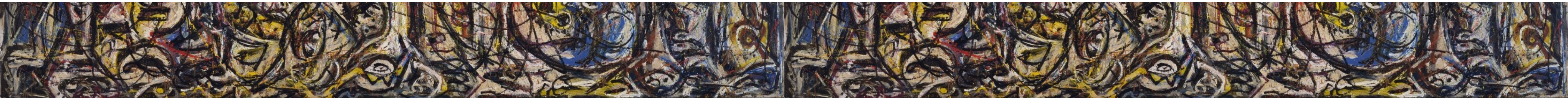
- Immensely useful, can be used to build $i\mathcal{O}$ (and much more!). Need maps of poly degree.



# Bootsrapping Based Constructions: Reduce, Reduce, Reduce



- What is the minimum functionality needed for $i\mathcal{O}$?
- How much can we "clean up" assumptions?
- Much progress



# Multilinear Maps to $i\mathcal{O}$

$i\mathcal{O}$

[GGHRSW13,BR14,BGKPS14,PST14, GLSW14,AGIS14,Zim15,AB15,GMS16,MSZ16]

**poly-degree**

**multilinear maps**

Slide Credit: Rachel Lin

# Multilinear Maps to $i\mathcal{O}$

$i\mathcal{O}$

[GGHRSW13,BR14,BGKPS14,PST14,
GLSW14,AGIS14,Zim15,AB15,GMS16,MSZ16]

**poly-degree**
**multilinear maps**

**Simplify functionality**

**constant-deg**

**[Lin16,AS17]**

**Multilinear maps**
**(but messy assumptions)**

Slide Credit: Rachel Lin

# Multilinear Maps to $i\mathcal{O}$

$i\mathcal{O}$

[GGHRSW13,BR14,BGKPS14,PST14,
GLSW14,AGIS14,Zim15,AB15,GMS16,MSZ16]

**poly-degree**
**multilinear maps**

**Simplify**
**functionality**

**[Lin16,AS17]**

**[LV16]**

**constant-deg**
**Multilinear maps**
**(but messy assumptions)**

**Simplify**
**Assumption**

**constant-deg**
**Multilinear maps**
**w/ nice assumptions**

Slide Credit: Rachel Lin

# Multilinear Maps to *iO*

*iO*

[GGHRSW13,BR14,BGKPS14,PST14,
GLSW14,AGIS14,Zim15,AB15,GMS16,MSZ16]

poly-degree
multilinear maps

Simplify
functionality

constant-deg
Multilinear maps
(but messy assumptions)

[Lin16,AS17]

Simplify
Assumption

constant-deg
Multilinear maps
w/ nice assumptions

[LV16]

Reduce
Degree

Deg 3 multilinear maps
w/ nice assumption

[Lin17, LT17]

Slide Credit: Rachel Lin

# Multilinear Maps to *iO*

Deg 3 Barrier: Need to generate randomness obliviously, needs degree at least 3 [LV18,BBKK18]

*Open #1 : Construct secure multilinear maps of degree > 2*

Have: Degree 2 (bilinear) maps from elliptic curve groups

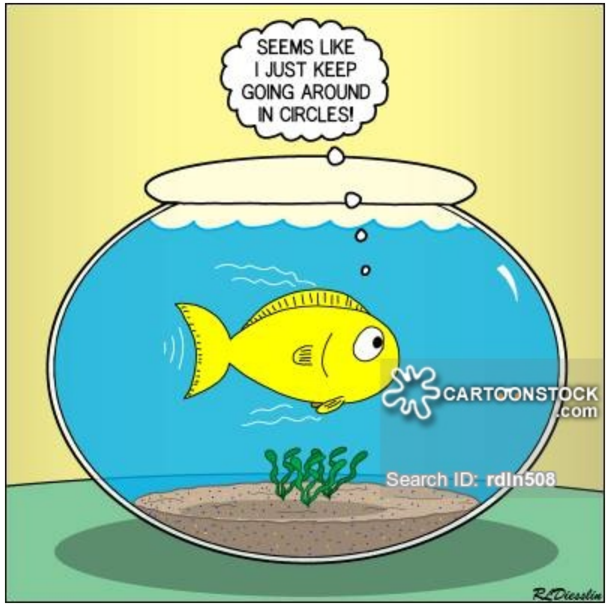
Big Open Problem

Desired: Deg 3+ maps, iO-land

# Can we base $iO$ on anything else?

# Can we base $i\mathcal{O}$ on anything else?

- *Functional Encryption* supporting computation of degree ≥ 3 polynomials [AJ15, BV15, Lin16, LV16, Lin17, AS17, LT17]
- Should be good news except…..
  - All constructions of degree 3 functional encryption themselves based on multilinear maps ☹



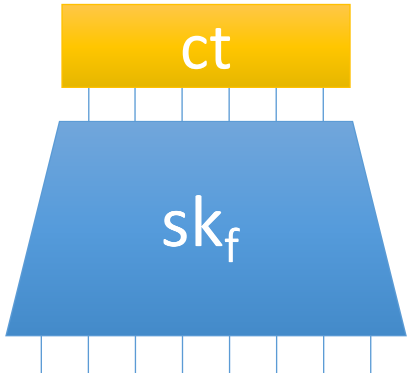
# Functional Encryption (FE) [SW05,BSW11]

(mpk, msk) ← Setup($1^n$)

Enc(mpk, x):


Kgen(msk, f):


Dec( $sk_C$, ct ):




y = f(x)

# Selective IND Security

MSK, MPK

$x_0, x_1$ ←

$Enc(x_b)$ →

$MPK$ →

$f$ ←

$SK_f$ →

$Bit\ b^*$ ←

Repeat Q times

Require $f(x_0)=f(x_1)$

Adversary wins if b = b*

Ciphertext size should be sublinear in Q to imply $iO$ [AJ15,BV15]

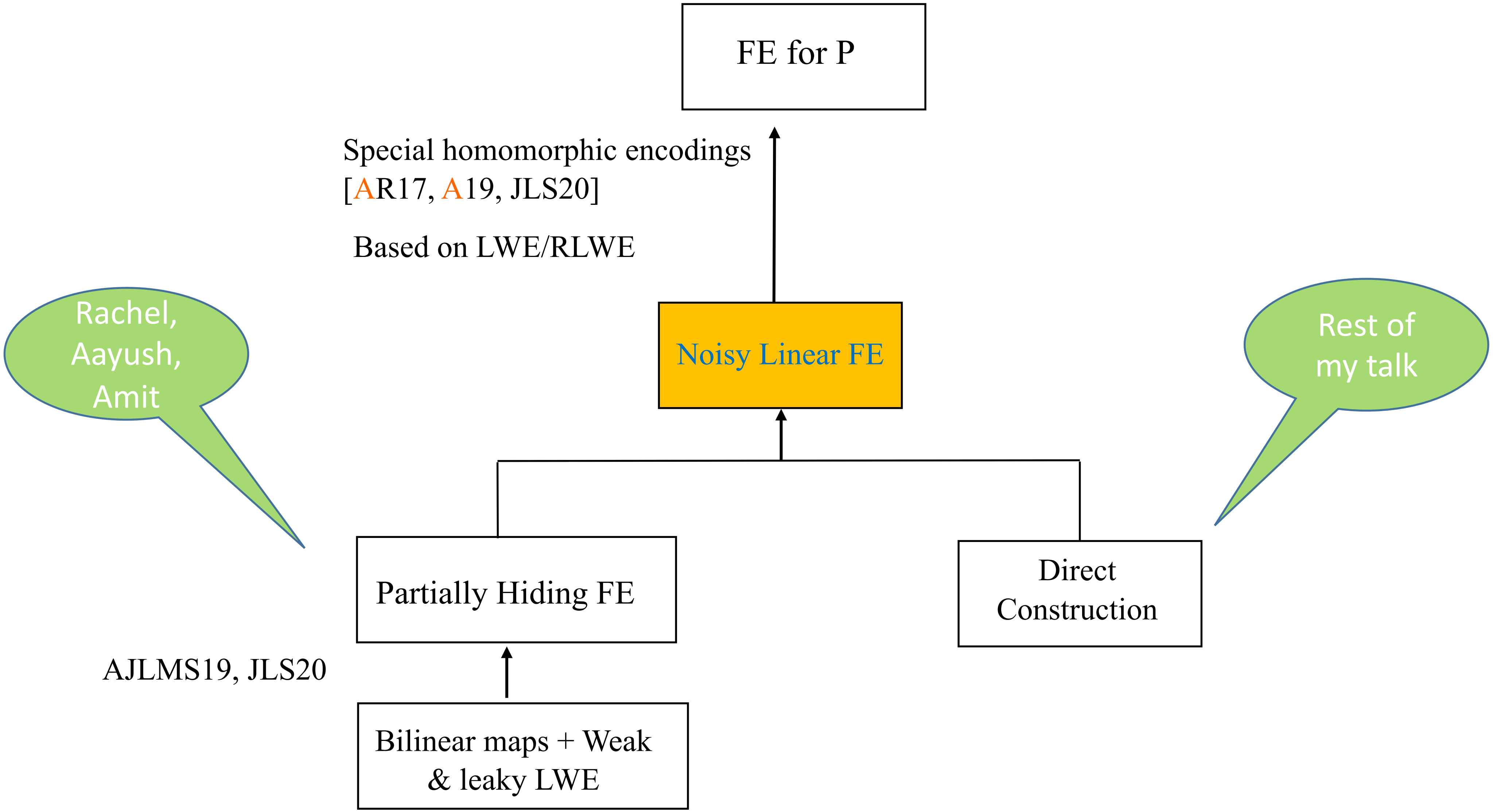
# $i\mathcal{O}$ from Functional Encryption

- Long sequence of works: Functional Encryption supporting computation of constant degree polynomials [AJ15,BV15,Lin17,AS17,LT17]
- Symmetric key FE suffices [BNPW16,KNT17]
- New Abstraction [A19]: Noisy linear functional encryption

Weaker than functional encryption for degree 3 polynomials!

Theorem (A19) : Special fully homomorphic encryption (LWE)
+ (Sel. IND Secure) Noisy Linear FE ➔ $i\mathcal{O}$

# Noisy Linear FE to *iO*



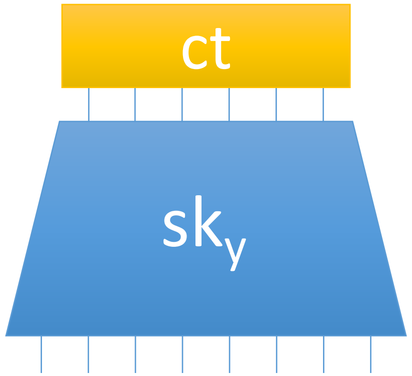
# ~~Noisy~~ Linear Functional Encryption [ABDP15,ALS16]

Let x, y ∈ $R^n$

Enc(mpk, x):


Kgen(msk, y):


Dec( $sk_y$, ct ):




<x, y> ∈ $R$

Can be constructed from standard assumptions – DDH, LWE, QR

# Noisy Linear Functional Encryption

Add noise to output!

Enc(x), Keygen(y), Decrypt to get <x,y> <u>plus noise</u>

- Where does noise come from?
- What security properties does it need to satisfy?
- Isn’t this high degree computation? Going in circles ?

# Noisy Linear Functional Encryption

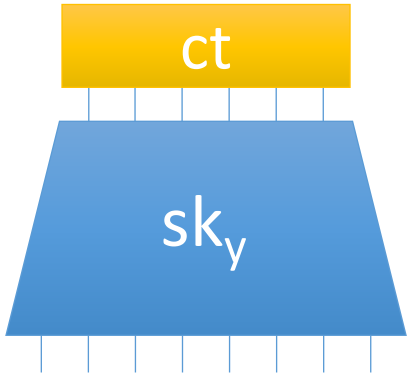
# Noisy Linear Functional Encryption [A19]

Let x, y ∈ $R^n$

Enc(msk, x):


Keygen(msk, y):


Dec( $sk_y$, ct ):




<x, y> + noise ∈ $R$

Noise must:

- be bounded by Bd
- satisfy weak pseudorandomness

# Selective IND Security

$\{ x_i^0, x_i^1 \}$

$\{Enc\ ( x_i^b)\}$

$y_j$

$SK(y_j)$

Require

$$|\langle y_j, x_i^0 - x_i^1 \rangle| < B$$

$$\forall i, j \in [Q]$$

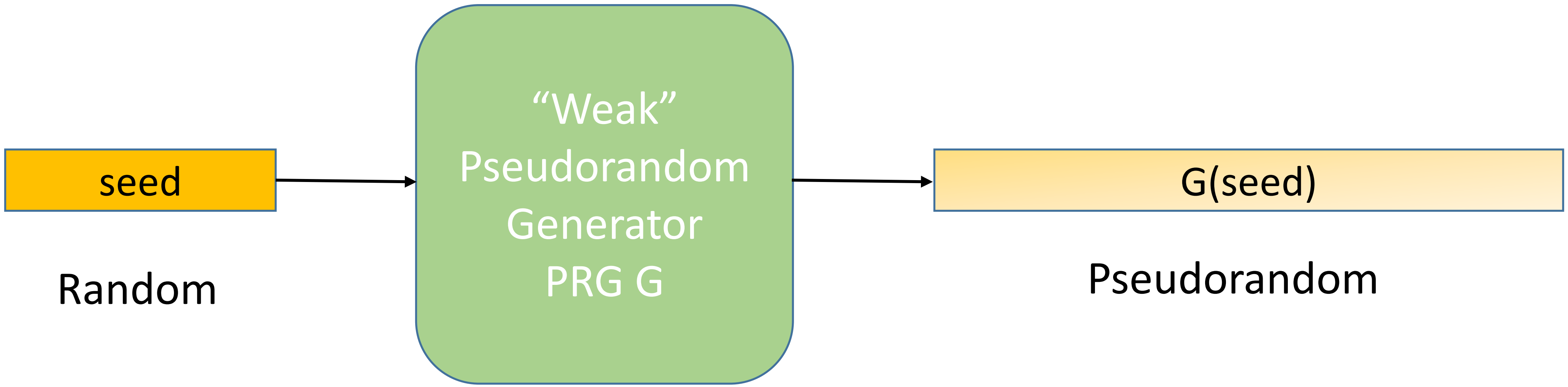
*Bit b**

Adversary wins if b = b*

Ciphertext size should be sublinear in Q to imply $iO$ [A19]

# Where does noise come from?

## Use PRG to compute noise




- Represent G as polynomial.
- Use FE to compute G(seed) and add it to output

# Key new observation: Old grandma advice!

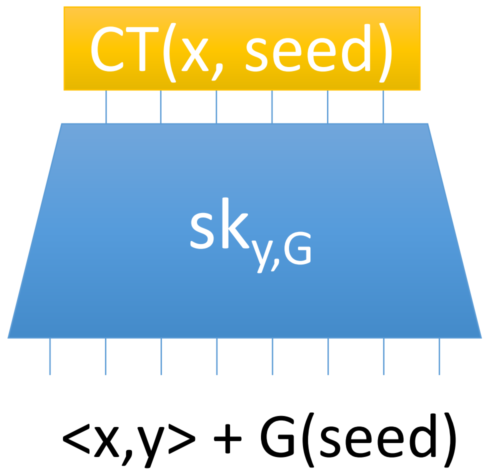
If you cannot have
what you want, you
must learn to want
what you can have

# Key new observation: Relax requirement on correctness!

If you cannot compute what you can use, you must learn to use what you can compute

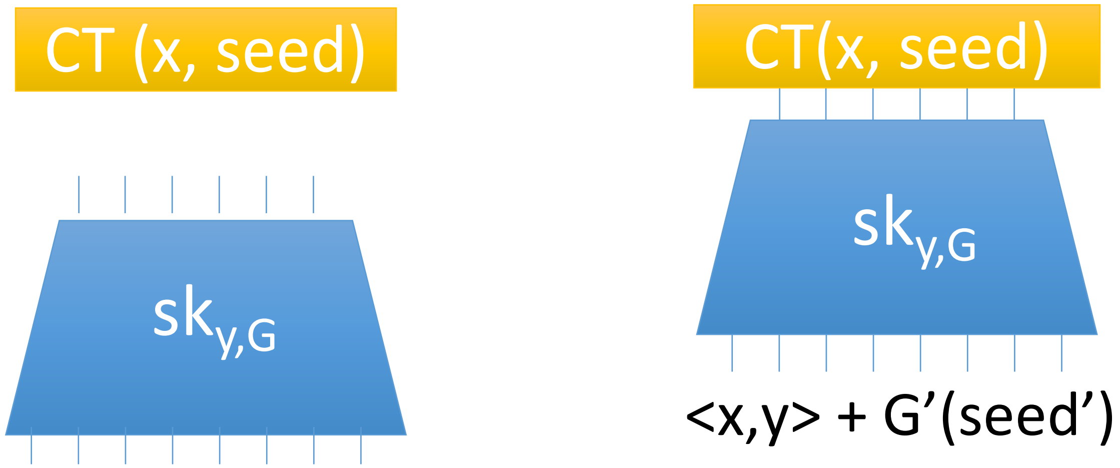
# Use PRG to compute noise

- Use FE to compute <x, y> + G(seed).




- Only <x,y> needs to be correct!
- Precise value of G(seed) not important

# A key new observation: Relax requirement on correctness!




- So far: Assume polynomial is PRG and insist on computing it exactly
- Here: Compute whatever can be computed and check if it can satisfy PRG like properties

# Permits New Direct Constructions

- Extend LWE based Linear FE of ALS16 to Noisy Linear FE using new hardness conjectures on lattices.
- Very different from multilinear map assumptions
- May be more robust. May be post-quantum.
- Much simpler to analyse than mmap based direct constructions: no need for straddling sets, Kilian randomization etc used by all prior work

First construction of $iO$ without <u>any</u> maps.


Construction

# Ring Learning with Errors Problem

Let ring $R_q = Z_q[x]/ < x^n + 1 >$

**DISTRIBUTION 1**

Sample s uniformly in $R_q$

$a_1$ , $b_1 = a_1 s + err_1$

$a_2$ , $b_2 = a_2 s + err_2$

$\vdots$

$a_m$ , $b_m = a_m s + err_m$

$a_i$ uniform $\in R_q$ , $e_i \sim \varphi \in R$

vs

**DISTRIBUTION 2**

$a'_1$ , $b'_1$

$a'_2$ , $b'_2$

$\vdots$

$a'_m$ , $b'_m$

$a'_i$, $b'_i$ uniform $\in R_q$

# Regev Public Key Encryption

Recall: Finding short $\vec{e}$ such that $<\vec{a};\vec{e}>=u$ is hard

- SK : $\vec{e}$   PK : $\vec{a},u$

Pseudorandom
By R-LWE

- Encrypt (PK, x) :

$$\vec{c}_0 = \vec{a}\cdot s + 2\cdot \vec{err}_1$$

$$c_1 = u\cdot s + 2\cdot err_2 + x$$

Small only if
e is small

- Decrypt (SK) :

$$\begin{aligned} c_1 - \langle \vec{e};\vec{c}_0\rangle &= u\cdot s + 2\cdot err_2 + x - u\cdot s - \langle \vec{e};\vec{err}_1\rangle \\ &= x + 2\cdot err \\ &= x \mod 2 \end{aligned}$$

# Linear Functional Encryption [ALS16]

MSK: $\vec{e}_1, \ldots \vec{e}_\ell$ (short)

PK: $\vec{a},\ \vec{u} = (u_1, \ldots, u_\ell)$

where $\langle \vec{a};\ \vec{e}_i \rangle = u_i \in R_q$

# Linear Functional Encryption [ALS16]

MSK: $\vec{e}_1, \ldots \vec{e}_\ell$ (short)

PK: $\vec{a},\ \vec{u} = (u_1, \ldots, u_\ell)$

where $\langle \vec{a};\ \vec{e}_i \rangle = u_i \in R_q$

Enc( PK, x):

$$\vec{c}_0 = \vec{a} \cdot s + 2 \cdot er\vec{r}_0$$

$$\vec{c}_1 = \vec{u} \cdot s + 2 \cdot er\vec{r}_1 + \vec{x}$$

# Linear Functional Encryption [ALS16]

MSK: $\vec{e}_1, \ldots \vec{e}_\ell$ (short)

PK: $\vec{a},\ \vec{u} = (u_1, \ldots, u_\ell)$

where $\langle \vec{a};\ \vec{e}_i \rangle = u_i \in R_q$

Enc( PK, x):

$$\vec{c}_0 = \vec{a} \cdot s + 2 \cdot er\vec{r}_0$$

$$\vec{c}_1 = \vec{u} \cdot s + 2 \cdot er\vec{r}_1 + \vec{x}$$

KeyGen( MSK, y):

$$\sum_{i \in [\ell]} y_i\ \vec{e}_i$$

# Linear Functional Encryption [ALS16]

MSK: $\vec{e}_1, \ldots \vec{e}_\ell$ (short)

PK: $\vec{a},\ \vec{u} = (u_1, \ldots, u_\ell)$

where $\langle \vec{a};\ \vec{e}_i \rangle = u_i \in R_q$

Enc( PK, x):

$$\vec{c}_0 = \vec{a} \cdot s + 2 \cdot e\vec{rr}_0$$

$$\vec{c}_1 = \vec{u} \cdot s + 2 \cdot e\vec{rr}_1 + \vec{x}$$

KeyGen( MSK, y):

$$\textstyle\sum_{i \in [\ell]} y_i\ \vec{e}_i$$

Decrypt:

$$\Big(\sum_{i \in [\ell]} y_i\ \vec{e}_i\Big)^\top \cdot \vec{c}_0 = \Big(\sum_{i \in [\ell]} y_i\ \vec{u}_i\Big) \cdot s + 2 \cdot err$$

$$-\quad \vec{y}^T \vec{c}_1 = \Big(\sum_{i \in [\ell]} y_i\ u_i\Big) \cdot s + 2 \cdot err + \langle \vec{x};\ \vec{y} \rangle$$

$$=\quad \langle \vec{x},\ \vec{y} \rangle + 2 \cdot err$$

# Wait a minute….

- Decryption reveals $\langle \vec{x},\ \vec{y} \rangle + 2 \cdot err$ : inner product + noise
- Isn't this noisy linear FE already?

Noise not pseudorandom

Noise is learnt fully after sufficient key requests!

# Adding Noise to Linear FE

- Starting point idea: Linear FE computes $\langle \vec{x},\ \vec{y} \rangle$ where $\vec{x}, \vec{y} \in R^{\ell}$
- Add dummy co-ordinate $x[\ell + 1] = \text{noise}, \quad y[\ell + 1] = 1$
- Now output $\langle \vec{x},\ \vec{y} \rangle + \text{noise}$
- Repeat Q times, once for each key request

Satisfies security, violates succinctness
CT size grows with Q

# Can we compute encodings of noise “on the fly”?

- Polynomial for computing noise must be degree at least 3 [LV18, BBKK18]

- Recall: Do not have FE for degree 3 polynomials without mmaps

- Is approximate computation easier?

# Is approximate computation easier?
# Or, Enter NTRU

Let $R = Z[x]/\langle x^n + 1\rangle$, $p_1 < p_2$ primes, $R_{p_1} = R/(p_1 \cdot R)$, $R_{p_2} = R/(p_2 \cdot R)$

$$\boxed{\text{Want to compute } d = h \cdot s + p_1 \cdot err + noise}$$

"noise" is message!

For $i \in \{1, \ldots, w\}$, sample $f_{1i}, f_{2i}$ and $g_1, g_2$ from a discrete Gaussian over ring $R$. Set

$$h_{1i} = \frac{f_{1i}}{g_1}, \quad h_{2j} = \frac{f_{2j}}{g_2} \in R_{p_2} \ \forall \ i, j \in [w]$$

Assume these look random.
Note difference from NTRU: Reusing denominator!

# RLWE with Structured Noise

Discrete Gaussian

$$\text{Want to compute } d = h \cdot s + p_1 \cdot err + noise$$

**Sample**

$$e_{1i} \leftarrow \widehat{\mathcal{D}}(\Lambda_2), \quad \text{where } \Lambda_2 \triangleq g_2 \cdot R. \ \text{Let } e_{1i} = g_2 \cdot \xi_{1i} \in \text{small},$$

$$e_{2i} \leftarrow \widehat{\mathcal{D}}(\Lambda_1), \quad \text{where } \Lambda_1 \triangleq g_1 \cdot R. \ \text{Let } e_{2i} = g_1 \cdot \xi_{2i} \in \text{small},$$

**Recall**

$$h_{1i} = \frac{f_{1i}}{g_1}, \quad h_{2j} = \frac{f_{2j}}{g_2}$$

**We have that:** $h_{1i} \cdot e_{2j} = f_{1i} \cdot \xi_{2j}, \quad h_{2j} \cdot e_{1i} = f_{2j} \cdot \xi_{1i} \quad \in$ small

# RLWE with Structured Noise

$\boxed{\text{Want to compute } d = h \cdot s + p_1 \cdot err + noise}$

We showed: $h_{1i} \cdot e_{2j} = f_{1i} \cdot \xi_{2j}, \quad h_{2j} \cdot e_{1i} = f_{2j} \cdot \xi_{1i} \quad \in$ small

Compute encodings of "PRG seed" :

$$d_{1i} = \boxed{h_{1i}} \cdot t_1 + p_1 \cdot \boxed{e_{1i}} \in R_{p_2}$$
$$d_{2i} = \boxed{h_{2i}} \cdot t_2 + p_1 \cdot \boxed{e_{2i}} \in R_{p_2}$$

Multiply encodings:

$d_{1i} \cdot d_{2j} = (h_{1i} \cdot h_{2j}) \cdot (t_2\ t_2) + p_1 \cdot$ noise

As desired!

where noise $= p_1 \cdot \left(f_{1i} \cdot \xi_{2j} \cdot t_1 + f_{2j} \cdot \xi_{1i} \cdot t_2 + p_1 \cdot g_1 \cdot g_2 \cdot \xi_{1i} \cdot \xi_{2j}\right) \in$ small

# RLWE with Structured Noise

Noise lives in an ideal that "cancels" large term in RLWE sample!

Extends to higher degree!

"Theorem": Its easy to make noise!


Description oversimplified. Please see paper [A19]

# What about security?

# Security of NLinFE

- Some analysis in [A19]
- Proof from clumsy assumption in overly weak security game
  - Adversary only gets single ciphertext
- Security based on inability to find attacks ☹

But…

- Much simpler to analyse than previous direct constructions
- Minimises part which depends on heuristics
- Uses no maps: completely different design template
- Post Quantum?

# Rigorous Cryptanalysis

- Follow-up [AP20]: Two attacks, and a fix
- Concrete parameters suggested
- New cryptanalytic techniques
- Security on much firmer footing

Hope: Inspires new candidates!

# Attack 1: Multiple Ciphertext Attack

Structured noise annihilates large term not only in its own ciphertext but also in other ciphertexts

Consider large terms in

$$d_{1i}d'_{2j} = (h_{1i}h_{2j}) \cdot (t_1 t'_2) + p_1 \cdot \text{small}$$

$$d_{2j}d'_{1i} = (h_{2j}h_{1i}) \cdot (t_2 t'_1) + p_1 \cdot \text{small}$$

Same label computed in two different ways!

Consider secrets in

$$d_{1i}d_{2j} = (h_{1i}h_{2j}) \cdot (t_1 t_2) + p_1 \cdot \text{small}$$

$$d_{2j'}d_{1i'} = (h_{2j'}h_{1i'}) \cdot (t_2 t_1) + p_1 \cdot \text{small}.$$

Now, secrets same but labels are changing

# Attack 1: Multiple Ciphertext Attack

Legal Ciphertexts:

$$d_{1i} = h_{1i} \cdot t_1 + p_1 \cdot e_{1i}$$
$$d_{1i'} = h_{1i'} \cdot t_1 + p_1 \cdot e_{1i'}$$
$$d'_{1i} = h_{1i} \cdot t'_1 + p_1 \cdot e'_{1i}$$
$$d'_{1i'} = h_{1i'} \cdot t'_1 + p_1 \cdot e'_{1i'}$$

Label changes with index of element in ciphertext, secret changes with ciphertext.

Can vary independently

Can be expressed as:

$$\begin{pmatrix} d_{1i} & d_{1i'} \\ d'_{1i} & d'_{1i'} \end{pmatrix} = \begin{pmatrix} t_1 \\ t'_1 \end{pmatrix} \cdot \begin{pmatrix} h_{1i} & h_{1i'} \end{pmatrix} + p_1 \cdot \begin{pmatrix} e_{1i} & e_{1i'} \\ e'_{1i} & e'_{1i'} \end{pmatrix}.$$

# RLWE with correlated noise

Distinguish many samples

$$\left( \underbrace{\frac{f_{1i}}{g_1}}_{a} \cdot \underbrace{t_1[j]}_{s} + \underbrace{g_2 \cdot e_{1i}[j]}_{e}, \quad \frac{f_{2i}}{g_2} \cdot t_2[j] + g_1 \cdot e_{2i}[j] \right) \in R_q^2$$

from uniform in $R_q^2$ where $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ .

Everything in blue is small. We can ask to vary $i$ or $j$.

**Remark:** even if $g_1$ is small, $\frac{1}{g_1} \mod q$ is not.

# Attacking RLWE with correlated noise

Let $$b_{1i}[j] = \frac{f_{1i}}{g_1} \cdot t_1[j] + g_2 \cdot e_{1i}[j]$$

$$B_1 = \begin{pmatrix} b_{11}[1] & b_{11}[2] \\ b_{12}[1] & b_{12}[2] \end{pmatrix} = \frac{1}{g_1} A + g_2 E$$

with $A$ of rank 1.

Using linearity of determinant:

$$\det(B_1) = \frac{1}{g_1^2} \underbrace{\det(A)}_{0} + \frac{g_2}{g_1} \det \begin{pmatrix} A_{1,1} & B_{1,2} \\ A_{2,1} & B_{2,2} \end{pmatrix} + \frac{g_2}{g_1} \det \begin{pmatrix} B_{1,1} & A_{1,2} \\ B_{2,1} & A_{2,2} \end{pmatrix} + {g_2}^2 \det(B)$$

$$= \frac{g_2}{g_1} \cdot \text{small}$$

# Attacking RLWE with correlated noise

We have seen: $\det(B_1) = \frac{g_2}{g_1} \cdot \mathbf{small} \bmod q$

By symmetry: $\det(B_2) = \frac{g_1}{g_2} \cdot \mathbf{small} \bmod q$

Multiplying these, we obtain:

$$\det(B_1) \cdot \det(B_2) = \mathbf{small} \bmod q$$

Distinguishing attack!

# Attack 2: Use noise term from NLinFE

The adversary can honestly play the following game

| $\mathcal{C}$ | | $\mathcal{A}$ |
| --- | --- | --- |
| $b \leftarrow \{0,1\}$ | | choose $\mu$ small |
| | $\xleftarrow{\quad \mu \quad}$ | |
| compute large noise $N$ | | |
| if $b = 0$ | | repeat |
| $x = N$ | | |
| otherwise | | |
| $x = N + \mu$ | $\xrightarrow{\quad x \quad}$ | |
| | | guess $b$ |

# Using Structure of N

$$
\begin{aligned}
N = \textstyle\sum_{\ell,i,j} v_{ij}^{\times} \; \Big[ p_1 \cdot \Big( & p_1 \cdot (g_2^\ell \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^\ell \cdot \tilde{\xi}_{2j}^{\ell}) \\
& + p_0 \cdot (g_2^\ell \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^\ell \cdot \xi_{2j}^\ell + g_2^\ell \cdot \xi_{1i}^\ell \cdot g_1^\ell \cdot \tilde{\xi}_{2j}^{\ell}) \\
& + (f_{1i}^\ell \cdot t_1 \cdot \tilde{\xi}_{2j}^{\ell} + f_{2j}^\ell \cdot t_2 \cdot \tilde{\xi}_{1i}^{\ell}) \Big) \\
+ p_0 \cdot \Big( p_0 \cdot (g_2^\ell \cdot \xi_{1i}^\ell \cdot g_1^\ell \cdot \xi_{2j}^\ell) + (f_{1i}^\ell \cdot t_1 \cdot \xi_{2j}^\ell + \cdot f_{2j}^\ell \cdot t_2 \cdot \xi_{1i}^\ell) \Big) \Big]
\end{aligned}
$$

# Using Structure of N

$$
\begin{aligned}
N = \textstyle\sum_{\ell,i,j} v^{\times}_{ij} \Big[ & {\color{red}p_1} \cdot \Big( {\color{red}p_1} \cdot (g^\ell_2 \cdot \tilde{\xi}^{\ell}_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^{\ell}_{2j}) \\
& \quad + {\color{red}p_0} \cdot (g^\ell_2 \cdot \tilde{\xi}^{\ell}_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j} + g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \tilde{\xi}^{\ell}_{2j}) \\
& \quad + (f^\ell_{1i} \cdot t_1 \cdot \tilde{\xi}^{\ell}_{2j} + f^\ell_{2j} \cdot t_2 \cdot \tilde{\xi}^{\ell}_{1i}) \Big) \\
& + {\color{red}p_0} \cdot \Big( {\color{red}p_0} \cdot (g^\ell_2 \cdot \xi^\ell_{1i} \cdot g^\ell_1 \cdot \xi^\ell_{2j}) + (f^\ell_{1i} \cdot t_1 \cdot \xi^\ell_{2j} + \cdot f^\ell_{2j} \cdot t_2 \cdot \xi^\ell_{1i}) \Big) \Big]
\end{aligned}
$$

Look at noise modulo $p_1^2$, $p_0 p_1$ , $p_1$ , $p_0^2$ and $p_0$ to recover different components.

# Using Structure of N

$$p_1^2 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell}$$

$$p_0 p_1 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot (g_2^{\ell} \cdot \tilde{\xi}_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell} + g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \tilde{\xi}_{2j}^{\ell})$$

$$p_1 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot (f_{1i}^{\ell} \cdot t_1 \cdot \tilde{\xi}_{2j}^{\ell} + f_{2j}^{\ell} \cdot t_2 \cdot \tilde{\xi}_{1i}^{\ell})$$

$$p_0^2 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot g_2^{\ell} \cdot \xi_{1i}^{\ell} \cdot g_1^{\ell} \cdot \xi_{2j}^{\ell}$$

$$p_0 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot (f_{1i}^{\ell} \cdot t_1 \cdot \xi_{2j}^{\ell} + \cdot f_{2j}^{\ell} \cdot t_2 \cdot \xi_{1i}^{\ell})$$

# Using Structure of N

$$p_1^2 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot g_2^{\ell} \cdot \tilde{\xi}^{\ell}_{1i} \cdot g_1^{\ell} \cdot \tilde{\xi}^{\ell}_{2j}$$

$$p_0 p_1 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot (g_2^{\ell} \cdot \tilde{\xi}^{\ell}_{1i} \cdot g_1^{\ell} \cdot \xi^{\ell}_{2j} + g_2^{\ell} \cdot \xi^{\ell}_{1i} \cdot g_1^{\ell} \cdot \tilde{\xi}^{\ell}_{2j})$$

$$p_1 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot (f^{\ell}_{1i} \cdot t_1 \cdot \tilde{\xi}^{\ell}_{2j} + f^{\ell}_{2j} \cdot t_2 \cdot \tilde{\xi}^{\ell}_{1i})$$

$$p_0^2 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot g_2^{\ell} \cdot \xi^{\ell}_{1i} \cdot g_1^{\ell} \cdot \xi^{\ell}_{2j}$$

$$p_0 \cdot \sum_{\ell,i,j} v_{ij}^{\times} \cdot (f^{\ell}_{1i} \cdot t_1 \cdot \xi^{\ell}_{2j} + \cdot f^{\ell}_{2j} \cdot t_2 \cdot \xi^{\ell}_{1i}) \boxed{+ (0 \text{ or } \mu)}$$

# Using Structure of N

$$\sum_{\ell,i,j} v^{\times}_{ij} \cdot (f^{\ell}_{1i} \cdot t_1 \cdot \xi^{\ell}_{2j} + \cdot f^{\ell}_{2j} \cdot t_2 \cdot \xi^{\ell}_{1i}) + (0 \text{ or } \mu)$$

Noise containing the challenge.

# Using Structure of N

$$\sum_{\ell,i,j} v_{ij}^{\times} \cdot (f_{1i}^{\ell} \cdot t_1 \cdot \xi_{2j}^{\ell} + \cdot f_{2j}^{\ell} \cdot t_2 \cdot \xi_{1i}^{\ell}) + (0 \text{ or } \mu)$$

Red: depends on the secret key, Blue: depends on the ciphertext, Black: fixed

Can make red and blue vary independently.

# Using Structure of N

Can be written as:

$$\langle \vec{a}, \vec{b} \rangle + (0 \text{ or } \mu)$$

Red: depends on the secret key, Blue: depends on the ciphertext

Can make $\vec{a}$ and $\vec{b}$ vary independently.

# Using Structure of N

$$\boxed{A}\ \boxed{B} + \left(0 \ \text{or}\ \boxed{(\mu_{ij})_{ij}}\ \right)$$

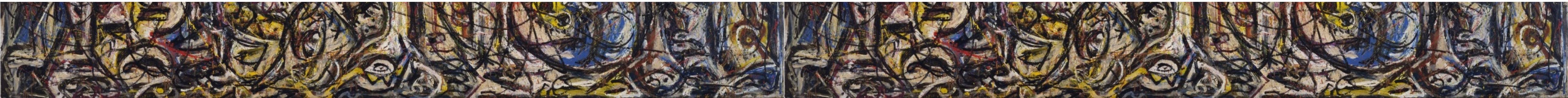
Can make $\vec{a}$ and $\vec{b}$ vary independently.

$\mu$ depends on both the secret key and the ciphertext.

# Using Structure of N

$$\boxed{A}\ \boxed{B} + \left(0 \ \text{or} \ \boxed{(\mu_{ij})_{ij}}\right)$$

To distinguish: compute rank
Full (or large) rank ➔ case μ, Small rank ➔ case 0



# Summing Up

- For fix to scheme, see paper [AP20]
- Takeaway: No fundamental security vulnerability
- Supports super-poly large output
- New design methodology

*Open:*

- *Proof from simple assumption?*
- *More candidates?*
- *Better efficiency?*

# Thank You for your attention!

Image Credits: Jackson Pollock, who solves similar problems in a different space!