Bar-Ilan University
Dept. of Computer Science

Fully Homomorphic Encryption

Shai Halevi – IBM Research

Based Mostly on [van-Dijk, Gentry, Halevi, Vaikuntanathan, EC 2010]

What is it?

Bar-Ilan University
Dept. of Computer Science

- **Homomorphic encryption: Can evaluate some functions on encrypted data**
  - E.g., from Enc(x), Enc(y) compute Enc(x+y)
- **Fully-homomorphic encryption: Can evaluate any function on encrypted data**
  - E.g., from Enc(x), Enc(y) compute Enc(x^3y-y^7+xy)

[redacted]

Part I

Somewhat Homomorphic Encryption (SHE)

- Evaluate low-degree polynomials on encrypted data

Motivating Application: Simple Keyword Search

- Storing an encrypted file F on a remote server
- Later send keyword w to server, get answer, determine whether F contains w
  - Trivially: server returns the entire encrypted file
  - We want: answer length independent of |F|

Claim: to do this, sufficient to evaluate low-degree polynomials on encrypted data
  - degree ~ security parameter

Protocol for keywork-search

Bar-Ilan University
Dept. of Computer Science

- File is encrypted bit by bit, $E(F_1)$ … $E(F_t)$
- Word has s bits $w_1 w_2 \ldots w_s$
- For i=1,2,…,t-s+1, server computes the bit $c_i = \prod_{j=1}^{s} (1 + w_j + F_{i+j-1}) \bmod 2$
  - $c_i = 1$ if $w = F_i F_{i+1} \ldots F_{i+s-1}$ (w found in position i) else $c_i = 0$
  - Each c_i is a degree-s polynomial in the F_i's
    - Trick from [Smolansky'93] to get degree-n polynomials, error-probability 2^{-n}
- Return n random subset-sums of the c_i's (mod 2) to client
  - Still degree-n, another 2^{-n} error

Computing low-degree polynomials on ciphertexts

Bar-Ilan University
Dept. of Computer Science

- **Want an encryption scheme (Gen, Enc, Dec)**
  - Say, symmetric bit-by-bit encryption
  - Semantically secure, E(0)≈E(1)
- **Another procedure: $C^*=Eval(f, C_1,\ldots C_t)$**
  - f is a binary polynomial in t variables, degree≤n
    - Represented as arithmetic circuit
  - The C_i's are ciphertexts
- **For any such f, and any $C_i=Enc(x_i)$ it holds that $Dec(Eval(f, C_1,\ldots C_t)) = f(x_1,\ldots,x_t)$**
  - Also |Eval(f,…)| does not depend on the "size" of f (i.e., # of vars or # of monomials, circuit-size)
  - That's called "compactness"

A Simple SHE Scheme

Bar-Ilan University
Dept. of Computer Science

- **Shared secret key: odd number p**
- **To encrypt a bit m:**
  - Choose at random small r, large q
  - Output c = pq + 2r + m (The "noise")
    - Ciphertext is close to a multiple of p
    - m = LSB of distance to nearest multiple of p
- **To decrypt c:**
  - Output m = (c mod p) mod 2
    = c – p • [[c/p]] mod 2
    = c – [[c/p]] mod 2
    = LSB(c) XOR LSB([[c/p]])

Noise much smaller than p

[[c/p]] is rounding of the rational c/p to nearest integer

Why is this homomorphic?

Bar-Ilan University
Dept. of Computer Science

- **Basically because:**
  - If you add or multiply two near-multiples of p, you get another near multiple of p…

Why is this homomorphic?

Bar-Ilan University
Dept. of Computer Science

- $c_1 = q_1 p + 2r_1 + m_1$, $c_2 = q_2 p + 2r_2 + m_2$

Distance to nearest multiple of p

- $c_1 + c_2 = (q_1 + q_2)p + 2(r_1 + r_2) + (m_1 + m_2)$
  - $2(r_1 + r_2) + (m_1 + m_2)$ still much smaller than p

  ➔ $c_1 + c_2 \bmod p = 2(r_1 + r_2) + (m_1 + m_2)$

- $c_1 \text{ x } c_2 = (c_1 q_2 + q_1 c_2 - q_1 q_2 p)p$
  $+ 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2$
  - $2(2r_1 r_2 + \ldots)$ still smaller than p

  ➔ $c_1 x c_2 \bmod p = 2(2r_1 r_2 + \ldots) + m_1 m_2$

Why is this homomorphic?

Bar-Ilan University
Dept. of Computer Science

- $c_1 = q_1 p + 2r_1 + m_1$, …, $c_t = q_t p + 2r_t + m_t$
- Let f be a multivariate poly with integer coefficients (sequence of +'s and x's)
- Let c = Eval(f, c_1, …, c_t) = f(c_1, …, c_t)

Suppose this noise is much smaller than p

  - $f(c_1, \dots, c_t) = f(m_1+2r_1, \dots, m_t+2r_t) + qp$
    $= f(m_1, \dots, m_t) + 2r + qp$

➔ $(c \bmod p) \bmod 2 = f(m_1, \dots, m_t)$

That's what we want!

How homomorphic is this?

Bar-Ilan University
Dept. of Computer Science

- **Can keep adding and multiplying until the "noise term" grows larger than p/2**
  - Noise doubles on addition, squares on multiplication
  - Initial noise of size ~ 2^n
  - Multiplying d ciphertexts ➔ noise of size ~2^{dn}
- **We choose r ~ 2^n, p ~ 2^{n^2} (and q ~ 2^{n^5})**
  - Can compute polynomials of degree ~n before the noise grows too large

Keeping it small

Bar-Ilan University
Dept. of Computer Science

- **Ciphertext size grows with degree of f**
  - Also (slowly) with # of terms
- **Instead, publish one "noiseless integer" N=pq**
  - For symmetric encryption, include N with the secret key and with every ciphertext
  - For technical reasons: q is odd, the q_i's are chosen from [q] rather than from $[2^{n^5}]$
- **Ciphertext arithmetic mod N**

➔Ciphertext-size remains always the same

Aside: Public Key Encryption

Bar-Ilan University
Dept. of Computer Science

Rothblum'11: Any homomorphic and compact symmetric encryption (wrt class *C* including linear functions), can be turned into public key

- Still homomorphic and compact wrt essentially the same class of functions *C*

- Public key: t random bits $m=(m_1 \ldots m_t)$ and their symmetric encryption $c_i=Enc_{sk}(m_i)$
  - t larger than size of evaluated ciphertext
- $NewEnc_{pk}$ (b): Choose random s s.t. <s,m>=b, use Eval to get $c^*=Enc_{sk}(\langle s,m \rangle)$
  - Note that s → c* is shrinking

Used to prove security

Security of our Scheme

Bar-Ilan University
Dept. of Computer Science

- **The approximate-GCD problem:**
  - Input: integers $w_0, w_1, \ldots, w_t$,
    - Chosen as $w_0=q_0p$, $w_i=q_ip + r_i$ (p and q_0 are odd)
    - $p \in_{\$} [0,P]$, $q_i \in_{\$} [0,Q]$, $r_i \in_{\$} [0,R]$ (with $R << P << Q$)
  - Task: find p
- **Thm: If we can distinguish Enc(0)/Enc(1) for some p, then we can find that p**
  - Roughly: the LSB of r_i is a "hard core bit"

➔ If approx-GCD is hard then scheme is secure

- **(Later: Is approx-GCD hard?)**

Hard-core-bit theorem

Bar-Ilan University
Dept. of Computer Science

A. The approximate-GCD problem:

- Input: $w_0=q_0p$, $\{w_i=q_ip+r_i\}$
  - $p\in_\$[0,P]$, $q_i\in_\$[0,Q]$, $r_i\in_\$[0,R]$ (with $R << P << Q$)
- Task: find p

B. The cryptosystem

labeled examples

challenge ciphertext

- Input: : $N=q_0p$, $\{m_j,\ c_j=q_jp+2\rho_j+m_j\}$, $c=qp+2\rho+m$
  - $p\in_\$[0,P]$, $q_i\in_\$[0,Q]$, $\rho_i\in_\$[0,R']$ (with $R' << P << Q$)
- Task: distinguish m=0 from m=1

Thm: Solving B ➔ solving A

- small caveat: R smaller than R'

Proof outline

Bar-Ilan University
Dept. of Computer Science

- Input: $w_0 = q_0 p$, $\{w_i = q_i p + r_i\}$
- Use the w_i's to form the c_j's and c
- Amplify the distinguishing advantage
  - From any noticeable ε to almost 1
  - This is where we need R'>R
- Use reliable distinguisher to learn q_0
  - Using the binary GCD procedure
- Finally $p = w_0/q_0$

From $\{w_i\}$ to $\{c_j, LSB(r_j)\}$

Bar-Ilan University
Dept. of Computer Science

- **We have $w_i=q_ip+r_i$, need $x_i=q_i'p+2\rho_i$**
  - Then we can add the LSBs to get $c_j = x_j + m_j$
- **Set $N=w_0$, $x_i=2w_i \bmod N$**
  - Actually $x_i=2(w_i+\rho_i) \bmod N$ with ρ_i random $< R'$
- **Correctness:**
  - The multipliers q_i, noise r_i, behave independently
    - As long as noise remain below p/2
  - $r_i+\rho_j$ distributed almost as ρ_j
    - $R'>R$ by a super-polynomial factor
  - $2\times q_i \bmod q_0$ is random in $[q_0]$

Amplify distinguishing advantage

Bar-Ilan University
Dept. of Computer Science

- **Given *any* integer $z=qp+r$, with $r<R$:**

  Set $c = [z + m + 2(\rho + \text{subsetSum}\{w_i\})] \bmod N$
  - For random $\rho<R'$, random bit m
- **For <u>*every*</u> z (with small noise), c is a nearly random ciphertext for $m+\text{LSB}(r)$**
  - $\text{subsetSum}(q_i\text{'s}) \bmod q_0$ almost uniform in $[q_0]$
  - $\text{subsetSum}(r_i\text{'s})+\rho$ distributed almost identically to ρ
- **For every $z=qp+r$, generate random ciphertexts for bits related to $\text{LSB}(r)$**

Amplify distinguishing advantage

Bar-Ilan University
Dept. of Computer Science

- **Given *any* integer z=qp+r, with r<R:**

  Set $c = [z + m + 2(\rho + \text{subsetSum}\{w_i\})] \bmod N$
  - For random $\rho < R'$, random bit m
- **For *every* z (with small noise), c is a nearly random ciphertext for m+LSB(r)**
  - A guess for c mod p mod 2 ➔ vote for r mod 2
- **Choose many random c's, take majority**

  Noticeable advantage for random c's
  ➔ Reliably computing r mod 2
  for every z with small noise

Reliable distinguisher ➔ The Binary GCD Algorithm

Bar-Ilan University
Dept. of Computer Science

- **From *any* $z=qp+r$ ($r<R'$) can get $r \bmod 2$**
  - Note: $z = q+r \bmod 2$ (since p is odd)
  - So ($q \bmod 2$) = ($r \bmod 2$) $\oplus$ ($z \bmod 2$)
- **Given z_1, z_2, both near multiples of p**

Binary-GCD:
  - Get $b_i := q_i \bmod 2$, if $z_1<z_2$ swap them
  - If $b_1=b_2=1$, set $z_1:=z_1-z_2$, $b_1:=b_1-b_2$
    - At least one of the b_i's must be zero now
  - For any $b_i=0$ set $z_i :=$ floor($z_i/2$)
    - new-q_i = old-$q_i/2$
  - Repeat until one z_i is zero, output the other

$z = (2s)p + r$
➔ $z/2=sp+r/2$
➔ floor($z/2$) = sp+floor($r/2$)

Binary GCD example (p=19)

Bar-Ilan University
Dept. of Computer Science

- $z_1 = 99 = 5\text{x}19 + 4\ (b_1=1)$
  $z_2 = 54 = 3\text{x}19 - 3\ (b_2=1)$
  - $z_1' = z_1 - z_2 = 45 = 2\text{x}19 + 7\ (b_1'=0)$
    $z_1'' = \text{floor}(z_1'/2) = 22 = 1\text{x}19 + 3$
- $z_1 = 54 = 3\text{x}19 - 3\ (b_1=1)$
  $z_2 = 22 = 1\text{x}19 + 3\ (b_2=1)$
  - $z_1' = z_1 - z_2 = 32 = 2\text{x}19 - 6\ (b_1'=0)$
    $z_1'' = z_1'/2 = 16 = 1\text{x}19 - 3$
- $z_1 = 22 = 1\text{x}19 + 3\ (b_1=1)$
  $z_2 = 16 = 1\text{x}19 - 3\ (b_2=1)$
  - $z_1' = z_1 - z_2 = 6 = 0\text{x}19 + 6$
    $z_1'' = z_1'/2 = 3 = 0\text{x}19 + 3$

Binary GCD example (p=19)

Bar-Ilan University
Dept. of Computer Science

- $z_1 = 16 = 1\text{x}19 - 3$ $(b_1=1)$
  $z_2 = 3 = 0\text{x}19 + 3$ $(b_2=0)$
  - $z_2'' = \text{floor}(z_2/2) = 1 = 0\text{x}19 + 1$
- $z_1 = 16 = 1\text{x}19 - 3$ $(b_1=1)$
  $z_2 = 1 = 0\text{x}19 + 1$ $(b_2=0)$
  - $z_2'' = \text{floor}(z_2/2) = 0$
- **Output** $16 = 1\text{x}19 - 3$
  - Indeed $1=\text{GCD}(5,3)$

The Binary GCD Algorithm

Bar-Ilan University
Dept. of Computer Science

- $z_i = q_i p + r_i$, $i=1,2$, $z' := \text{OurBinaryGCD}(z_1, z_2)$
  - Then $z' = GCD^*(q_1, q_2) \cdot p + r'$
  - For random q,q', Pr[GCD(q,q')=1] ~ 0.6

The odd part of the GCD

Binary GCD → learning q_0,p

Bar-Ilan University
Dept. of Computer Science

- **Try (say) z':= OurBinaryGCD(w_0,w_1)**
  - Hope that $z'=1\cdot p+r$
    - Else try again with OurBinaryGCD(z',w_2), etc.
- **One you have $z'=1\cdot p+r$, run OurBinaryGCD(w_0,z')**
  - GCD(q_0,1)=1, but the b_1 bits along the way spell out the binary representation of q_0
- **Once you learn q_0, $p=w_0/q_0$**

QED

Where we are

Bar-Ilan University
Dept. of Computer Science

- We proved: If approximate-GCD is hard then the scheme is secure
- Next: is approximate-GCD really hard?

Is Approximate-GCD Hard?

Bar-Ilan University
Dept. of Computer Science

- **Several lattice-based approaches for solving approximate-GCD**
  - Approximate-GCD is related to Simultaneous Diophantine Approximation (SDA)
    - Can use Lagarias'es algorithm to attack it
  - Studied in [Hawgrave-Graham01]
    - We considered some extensions of his attacks
- **These attacks run out of steam when $|q_i|>|p|^2$**
  - In our case $|p|\sim n^2$, $|q_i|\sim n^5 >> |p|^2$

Lagarias'es SDA algorithm

- **Consider the rows of this matrix B:**
  - They span dim-(t+1) lattice
- **$(q_0,q_1,\ldots,q_t) \times B$ is short**
  - 1^{st} entry: $q_0 R < Q \cdot R$
  - i^{th} entry ($i>1$): $q_0(q_i p+r_i)-q_i(q_0 p)=q_0 r_i$
    - Less than $Q \cdot R$ in absolute value
  - ➔ Total size less than $Q \cdot R \cdot \sqrt{t}$
    - vs. size ~$Q \cdot P$ (or more) for basis vectors
- **Hopefully we can find it with a lattice-reduction algorithm (LLL or variants)**

$$B = \begin{pmatrix} R & w_1 & w_2 & \ldots & w_t \\ & -w_0 & & & \\ & & -w_0 & & \\ & & & \ldots & \\ & & & & -w_0 \end{pmatrix}$$

Will this algorithm succeed?

- **Is $(q_0,q_1,\ldots,q_t) \times B$ the shortest in the lattice?**
  - Is it shorter than $\sqrt{t}\cdot\det(B)^{1/t+1}$? ← Minkowski bound
    - $\det(B)$ is small-ish (due to R in the corner)
  - Need $((QP)^t R)^{1/t+1} > QR$

    $\Leftrightarrow t+1 > (\log Q + \log P - \log R) / (\log P - \log R)$

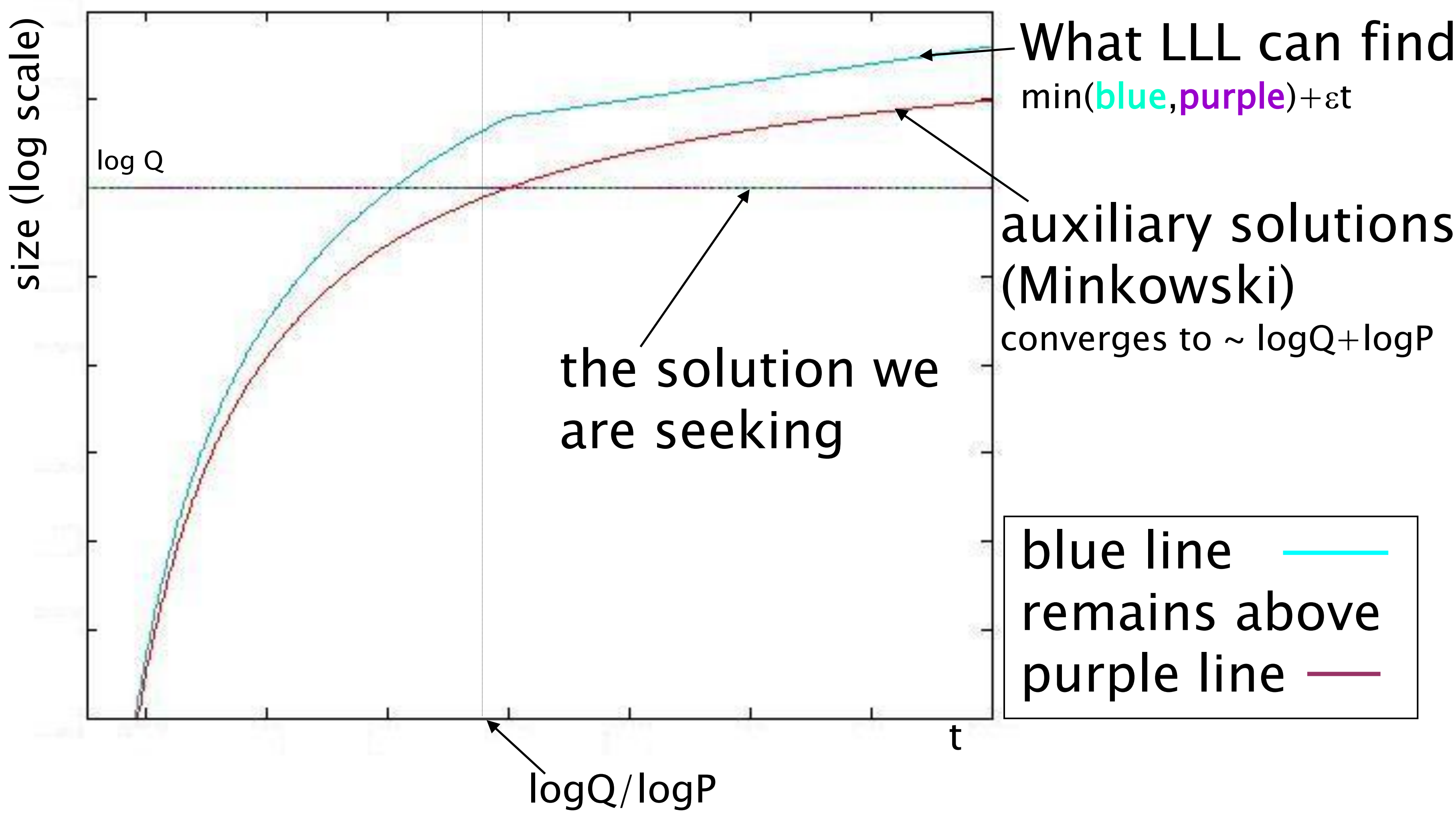
    $\sim \log Q/\log P$

$$\begin{pmatrix} R & w_1 & w_2 & \ldots & w_t \\ & -w_0 & & & \\ & & -w_0 & & \\ & & & \ldots & \\ & & & & -w_0 \end{pmatrix}$$

- **$\log Q = \omega(\log^2 P)$ ➔ need $t=\omega(\log P)$**
- **Quality of LLL & co. degrades with t**
  - Find vectors of size $\sim 2^{\varepsilon t}\cdot$shortest
  - $t=\omega(\log P)$ ➔ $2^{\varepsilon t}\cdot QR > \det(B)^{1/t+1}$
  - Contemporary lattice reduction not strong enough

Why this algorithm fails

Bar-Ilan University



Conclusions for Part I

Bar-Ilan University
Dept. of Computer Science

- **A Simple Scheme that supports computing low-degree polynomials on encrypted data**
  - Any fixed polynomial degree can be done
  - To get degree-d, ciphertext size must be $\omega(nd^2)$
- **Already can be used in applications**
  - E.g., the keyword-match example

- **Next we turn it into a fully-homomorphic scheme**

[redacted]

Part II

Fully Homomorphic Encryption

Bootstrapping [Gentry 09]

Bar-Ilan University
Dept. of Computer Science

- So far, can evaluate low-degree polynomials

x_1

x_2

…

x_t

$f(x_1, x_2, \ldots, x_t)$

Bootstrapping [Gentry 09]

Bar-Ilan University
Dept. of Computer Science

- So far, can evaluate low-degree polynomials

x_1
x_2
…
x_t

$f(x_1, x_2, \ldots, x_t)$

- Can eval $y=f(x_1,x_2\ldots,x_n)$ when x_i's are "fresh"
- But y is "evaluated ciphertext"
  - Can still be decrypted
  - But eval $Q(y)$ has too much noise

Bootstrapping [Gentry 09]

Bar-Ilan University
Dept. of Computer Science

- So far, can evaluate low-degree polynomials

x_1

x_2

…

x_t

$f(x_1, x_2, \ldots, x_t)$

- Bootstrapping to handle higher degrees:
- For a ciphertext c, consider $D_c(sk) = \mathrm{Dec}_{sk}(c)$
  - Hope: $D_c(*)$ has a low degree in sk
  - Then so are

$$A_{c_1,c_2}(sk) = \mathrm{Dec}_{sk}(c_1) + \mathrm{Dec}_{sk}(c_2)$$

and $$M_{c_1,c_2}(sk) = \mathrm{Dec}_{sk}(c_1) \times \mathrm{Dec}_{sk}(c_2)$$

Bootstrapping [Gentry 09]

Bar-Ilan University
Dept. of Computer Science

- Include in the public key also $\mathrm{Enc}_{pk}(sk)$

x_1 x_2

c_1 c_2

Requires "circular security"

sk_1

sk_2

...

sk_n

c

$\mathrm{M}_{c_1,c_2}(sk)$

$= \mathrm{Dec}_{sk}(c_1) \times \mathrm{Dec}_{sk}(c_2) = x_1 \times x_2$

- Homomorphic computation applied only to the "fresh" encryption of *sk*

Bootstrapping [Gentry 09]

Bar-Ilan University
Dept. of Computer Science

- Fix a scheme (Gen, Enc, Dec, Eval)
- For a class F of functions , denote
  - C_F = { Eval(f, $c_1,\ldots,c_t$) : f $\in F$, $c_i \in$ Enc(0/1) }
  - Encrypt some t bits and evaluate on them some f$\in F$
- Scheme *bootstrappable* if exists F for which:
  - Eval "works" for F
    - $\forall$ f $\in F$, $c_i \in$ Enc(x_i), Dec(Eval(f,$c_1,\ldots,c_t$)) = f($x_1,\ldots,x_t$)
  - Decryption + add/mult in F
    - $\forall$ $c_1, c_2 \in C_F$, $A_{c_1,c_2}(sk)$, $M_{c_1,c_2}(sk) \in F$

Thm: Circular secure
& Boostrappable
➔ Homomorphic for any func.

Is our SHE Bootstrappable?

Bar-Ilan University
Dept. of Computer Science

- $\text{Dec}_p(c) = \text{LSB}(c) \oplus \text{LSB}([[c/p]])$
  - We have $|c| \sim n^5$, $|p| \sim n^2$

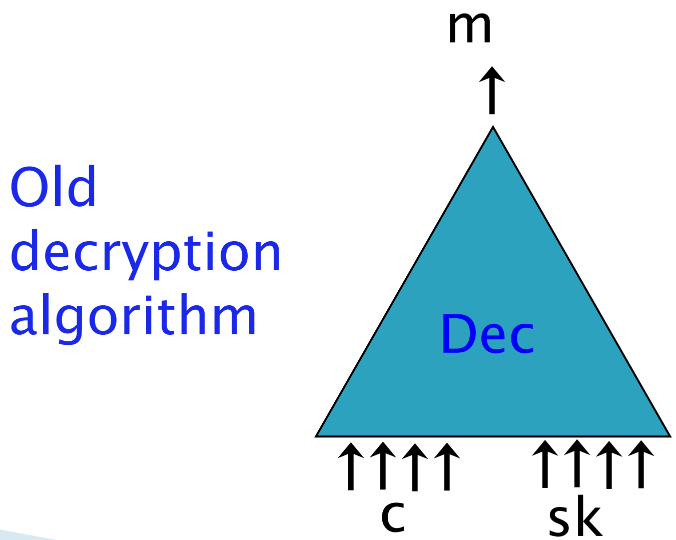
c/p, rounded to nearest integer

- Naïvely computing [[c/p]] takes degree $> n^5$
- Our scheme only supports degree ~ n
- Need to "squash the decryption circuit" in order to get a bootstrappable scheme
  - Similar techniques to [Gentry 09]

How to "Simplify" Decryption?

Bar-Ilan University
Dept. of Computer Science

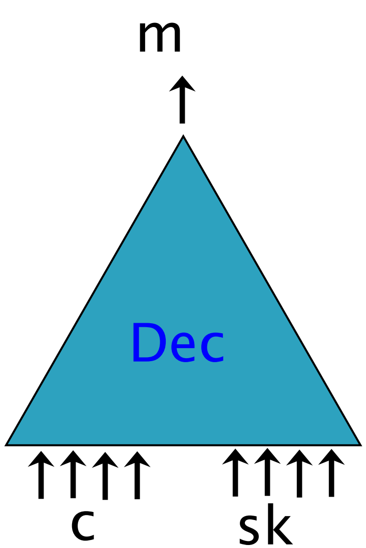
- **Add to public key another "hint" about sk**
  - Hint should not break secrecy of encryption
- **With hint, ciphertext can be publically post-processed, leaving less work for Dec**
- **Idea is used in server-aided cryptography.**

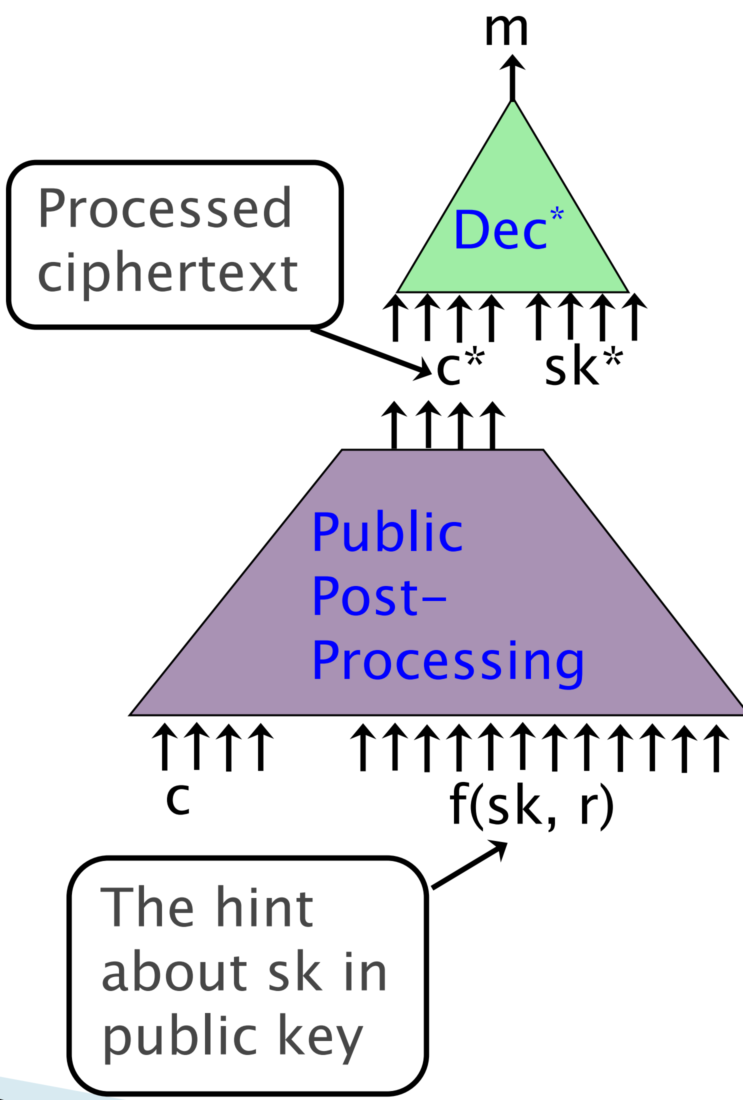


How to“simplify” decryption?

Bar-Ilan University
Dept. of Computer Science

Old decryption algorithm





New approach

Hint in pub key lets anyone post-process the ciphertext, leaving less work for Dec*

The New Scheme

Bar-Ilan University
Dept. of Computer Science

- Old secret key is the integer p
- Add to public key many "real numbers"
  - $d_1, d_2, \ldots, d_t \in [0,2)$ (with precision of ~|c| bits)
  - $\exists$ **sparse** S for which $\Sigma_{i \in S}\, d_i = 1/p \bmod 2$
- Post Processing: $\psi_i = c \times d_i \bmod 2,\ i=1,\ldots,t$
  - New ciphertext is $c^* = (c, \psi_1, \psi_2, \ldots, \psi_i)$
- New secret key is char. vector of S $(\sigma_1, \ldots, \sigma_t)$
  - $\sigma_i = 1$ if $i \in S$, $\sigma_i = 0$ otherwise
  - $c/p = c \times (\Sigma\, \sigma_i d_i) = \Sigma\, \sigma_i \Psi_i \bmod 2$

$$Dec^*(c^*) = c - [[\Sigma_i\, \sigma_i \Psi_i]] \bmod 2$$

The New Scheme

Bar-Ilan University
Dept. of Computer Science

- $Dec^*_\sigma(c^*) = LSB(c) \oplus LSB([[\Sigma_i \sigma_i \psi_i]])$

$\Psi_{1,0}$	$\Psi_{1,-1}$	…	$\Psi_{1,1-p}$	$\Psi_{1,-p}$	x σ_1
$\Psi_{2,0}$	$\Psi_{2,-1}$	…	$\Psi_{2,1-p}$	$\Psi_{2,-p}$	x σ_2
$\Psi_{3,0}$	$\Psi_{3,-1}$	…	$\Psi_{3,1-p}$	$\Psi_{3,-p}$	x σ_3
…	…	…		…	
$\Psi_{t,0}$	$\Psi_{t,-1}$	…	$\Psi_{t,1-p}$	$\Psi_{t,-p}$	x σ_t

$b = LSB([[\Sigma_i \sigma_i \psi_i]])$

The New Scheme

Bar-Ilan University
Dept. of Computer Science

- $\text{Dec*}_\sigma(c^*) = \text{LSB}(c) \oplus \text{LSB}([[\, \Sigma_i\ \sigma_i\psi_i\,]])$

σ_1	σ_1	…	0	σ_1	x σ_1
0	σ_2	…	σ_2	σ_2	x σ_2
σ_3	0	…	σ_3	0	x σ_3
…	…	…		…	
0	0	…	0	σ_t	x σ_t
b					

- Use grade-school addition to compute b

How to Add Numbers?

Bar-Ilan University
Dept. of Computer Science

- $Dec^*_\sigma(c^*) = LSB(c) \oplus \underbrace{LSB([[\Sigma_i \underbrace{\sigma_i\psi_i}_{a_i \in [0,2)}]])}_{b \in \{0,1\}}$

$a_{1,0}$	$a_{1,-1}$	…	$a_{1,1-p}$	$a_{1,-p}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,1-p}$	$a_{2,-p}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,1-p}$	$a_{3,-p}$
…	…	…		…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,1-p}$	$a_{t,-p}$
b				

The a_i's in binary: each $a_{i,j}$ is either σ_i or 0

- **Grade-school addition**
  - What is the degree of $b(\sigma_1,\ldots,\sigma_t)$?

Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$c_{1,0}$	$c_{1,-1}$	…	$c_{1,1-p}$	
$a_{1,0}$	$a_{1,-1}$	…	$a_{1,1-p}$	$a_{1,-p}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,1-p}$	$a_{2,-p}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,1-p}$	$a_{3,-p}$
…	…	…		…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,1-p}$	$a_{t,-p}$
				b_{-p}

Carry Bits

Result Bit

$c_{1,0}c_{1,-1} \ldots c_{1,1-p}\, b_{-p}$

$= \text{HammingWeight}(\text{Column}_{-p})$

$\text{mod } 2^{p+1}$

Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$c_{2,0}$	$c_{2,-1}$	…		
$c_{1,0}$	$c_{1,-1}$	…	$c_{1,1-p}$	
$a_{1,0}$	$a_{1,-1}$	…	$a_{1,1-p}$	$a_{1,-p}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,1-p}$	$a_{2,-p}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,1-p}$	$a_{3,-p}$
…	…	…		…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,1-p}$	$a_{t,-p}$
			b_{1-p}	b_{-p}

$c_{2,0}c_{2,-1} \dots c_{2,2-p}\, b_{1-p}$
$= \text{HammingWeight}(\text{Column}_{1-p}) \bmod 2^p$

Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$c_{p,0}$				
…	…			
$c_{2,0}$	$c_{2,-1}$	…		
$c_{1,0}$	$c_{1,-1}$	…	$c_{1,1-p}$	
$a_{1,0}$	$a_{1,-1}$	…	$a_{1,1-p}$	$a_{1,-p}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,1-p}$	$a_{2,-p}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,1-p}$	$a_{3,-p}$
…	…	…		…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,1-p}$	$a_{t,-p}$
	b_{-1}	…	b_{1-p}	b_{-p}

$c_{p,0}b_{-1}$ = HammingWgt(Col_{-1}) mod 4

Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$c_{p,0}$				
…	…			
$c_{2,0}$	$c_{2,-1}$	…		
$c_{1,0}$	$c_{1,-1}$	…	$c_{1,1-p}$	
$a_{1,0}$	$a_{1,-1}$	…	$a_{1,1-p}$	$a_{1,-p}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,1-p}$	$a_{2,-p}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,1-p}$	$a_{3,-p}$
…	…	…		…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,1-p}$	$a_{t,-p}$
b	b_{-1}	…	b_{1-p}	b_{-p}

- Express $c_{i,j}$'s as polynomials in the $a_{i,j}$'s

Small Detour: Elementary Symmetric Polynomials

Bar-Ilan University
Dept. of Computer Science

- Binary Vector $x = (x_1, \ldots, x_u) \in \{0,1\}^u$
- $e_k(x)$ = deg-k elementary symmetric polynomial
  - Sum of all products of k bits (u-choose-k terms)
- Dynamic programming to evaluate in time O(ku)
  - $e_i(x_1 \ldots x_j) = e_{i-1}(x_1 \ldots x_{j-1})x_j + e_i(x_1 \ldots x_{j-1})$ (for $i \le j$)

	Λ	x_1	x_1,x_2	…	$x_1 \ldots x_{u-1}$	$x_1 \ldots x_u$
e_0	1	1	1		1	1
e_1	0					
…				$e_i(x_1 \ldots x_j)$		
e_k	0					

The Hamming Weight

Bar-Ilan University
Dept. of Computer Science

Thm: For a vector $x = (x_1, \ldots, x_u) \in \{0,1\}^u$,
i'th bit of $W = HW(x)$ is $e_{2^i}(x) \bmod 2$

- Observe $e_{2^i}(x) = (W \text{ choose } 2^i)$
- Need to show: i'th bit of $W = (W \text{ choose } 2^i) \bmod 2$

- Say $2^k \leq W < 2^{k+1}$ (bit k is MSB of W), $W' = W - 2^k$
  - For $i < k$, $(W \text{ choose } 2^i) = (W' \text{ choose } 2^i) \quad \bmod 2$
  - For $i = k$, $(W \text{ choose } 2^k) = (W' \text{ choose } 2^k) + 1 \bmod 2$
- Then by induction over W
  - Clearly holds for $W = 0$
  - By above, if holds for $W' = W - 2^k$ then holds also for W

The Hamming Weight

Bar-Ilan University
Dept. of Computer Science

- Use identity $\binom{W}{2^i} = \sum_{j=0}^{2^i} \binom{W-2^k}{j}\binom{2^k}{2^i-j}$ (∗)
  - For r=0 or r=2^k we have (2^k choose r) = 1
  - For 0<r<2^k we have (2^k choose r) = 0 mod 2

Numerator has more powers of 2 than denominator

$$\binom{2^k}{r} = \frac{2^k}{r} \cdot \frac{(2^k-1)}{(r-1)} \cdots \frac{(2^k-r+1)}{1}$$

integer $= \binom{2^k-1}{r-1}$

- i<k: The only nonzero term in (∗) is j=2^i
- i=k: The only nonzero terms in (∗) are j=0 and j=2^k

QED

Back to Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$c_{4,0}$				
$c_{3,0}$	$c_{3,-1}$			
$c_{2,0}$	$c_{2,-1}$	$c_{2,-2}$		
$c_{1,0}$	$c_{1,-1}$	$c_{1,-2}$	$c_{1,-3}$	
$a_{1,0}$	$a_{1,-1}$	$a_{1,-2}$	$a_{1,-3}$	$a_{1,-4}$
$a_{2,0}$	$a_{2,-1}$	$a_{2,-2}$	$a_{2,-3}$	$a_{2,-4}$
…	…	…	…	…
$a_{t,0}$	$a_{t,-1}$	$a_{t,-2}$	$a_{t,-3}$	$a_{t,-4}$

Carry Bits

Input Bits

b

Goal:
compute the degree of the polynomial **b**($a_{i,j}$'s)

Back to Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$e_{16}(\ldots)$	$e_8(\ldots)$	$e_4(\ldots)$	$e_2(\ldots)$	
deg=1	deg=1	deg=1	deg=1	deg=1
deg=1	deg=1	deg=1	deg=1	deg=1
…	…	…	…	…
deg=1	deg=1	deg=1	deg=1	deg=1

Back to Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$e_8(\ldots)$	$e_4(\ldots)$	$e_2(\ldots)$		
deg=16	deg=8	deg=4	deg=2	
deg=1	deg=1	deg=1	deg=1	deg=1
deg=1	deg=1	deg=1	deg=1	deg=1
…	…	…	…	…
deg=1	deg=1	deg=1	deg=1	deg=1

Back to Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$e_4(\ldots)$	$e_2(\ldots)$			
deg=9	deg=5	deg=3		
deg=16	deg=8	deg=4	deg=2	
deg=1	deg=1	deg=1	deg=1	deg=1
deg=1	deg=1	deg=1	deg=1	deg=1
…	…	…	…	…
deg=1	deg=1	deg=1	deg=1	deg=1

Back to Grade School Addition

Bar-Ilan University
Dept. of Computer Science

$e_2(\ldots)$				
deg=9	deg=7			
deg=9	deg=5	deg=3		
deg=16	deg=8	deg=4	deg=2	
deg=1	deg=1	deg=1	deg=1	deg=1
deg=1	deg=1	deg=1	deg=1	deg=1
…	…	…	…	…
deg=1	deg=1	deg=1	deg=1	deg=1

Back to Grade School Addition

Bar-Ilan University
Dept. of Computer Science

deg=15				
deg=9	deg=7			
deg=9	deg=5	deg=3		
deg=16	deg=8	deg=4	deg=2	
deg=1	deg=1	deg=1	deg=1	deg=1
deg=1	deg=1	deg=1	deg=1	deg=1
…	…	…	…	…
deg=1	deg=1	deg=1	deg=1	deg=1

deg(b) = 16

<u>Claim</u>: with p bits of precision,
$\deg(b(a_{i,j})) \leq 2^p$

Our Decryption Algorithm

Bar-Ilan University
Dept. of Computer Science

- $Dec^*_\sigma(c^*) = LSB(c) \oplus LSB([[\Sigma_i \sigma_i \psi_i]])$

  ($LSB([[\Sigma_i \sigma_i \psi_i]]) = b \in \{0,1\}$; $\sigma_i \psi_i = a_i \in [0,2]$)

$a_{1,0}$	$a_{1,-1}$	…	$a_{1,1-p}$	$a_{1,-p}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,1-p}$	$a_{2,-p}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,1-p}$	$a_{3,-p}$
…	…	…		…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,1-p}$	$a_{t,-p}$
b				

The a_i's in binary: each $a_{i,j}$ is either σ_i or 0

- **degree(b) = 2^p**
  - We can only handle degree ~ n
  - Need to work with low precision, p ~ log n

Lowering the Precision

Bar-Ilan University
Dept. of Computer Science

- **Current parameters ensure "noise" < p/2**
  - For degree-2n polynomials with $< 2^{n^2}$ terms (say)
  - With $|r|=n$, need $|p|\sim 3n^2$
- **What if we want a somewhat smaller noise?**
  - Say that we want the noise to be $< p/2n$
  - Instead of $|p|\sim 3n^2$, set $|p|\sim 3n^2+\log n$
    - Makes essentially no difference

<u>Claim</u>: c has noise < p/2n
& sparse subset size ≤ n−1
➔ enough to keep precision of log n bits for the ψ_i's

Lowering the Precision

Bar-Ilan University
Dept. of Computer Science

<u>Claim</u>: $|S| \le n-1$ & c/p within $1/2n$ from integer
➔ enough to keep log n bits for the ψ_i's

<u>Proof</u>: ϕ_i = rounding of ψ_i to log n bits

- $|\phi_i - \psi_i| \le 1/2n$ ➔ $\sigma_i\phi_i = \begin{cases} \sigma_i\Psi_i & \text{if } \sigma_i=0 \\ \sigma_i\Psi_i \pm 1/2n & \text{if } \sigma_i=1 \end{cases}$

➔ $|\Sigma\sigma_i\phi_i - \Sigma\sigma_i\Psi_i| \le |S|/2n \le (n-1)/2n$

- $\Sigma\sigma_i\Psi_i = c/p$, within $1/2n$ of an integer

➔ $\Sigma\sigma_i\phi_i$ within $1/2n+(n-1)/2n=1/2$ of the same integer

➔ $[[\Sigma\sigma_i\phi_i]] = [[\Sigma\sigma_i\Psi_i]]$ QED

Bootstrappable, at last

Bar-Ilan University
Dept. of Computer Science

- $Dec^*_\sigma(c^*) = LSB(c) \oplus LSB([[\ \Sigma_i\ \sigma_i \phi_i\ ]])$

$\sigma_i\phi_i = a_i \in [0,2]$

$a_{1,0}$	$a_{1,-1}$	…	$a_{1,-\log n}$
$a_{2,0}$	$a_{2,-1}$	…	$a_{2,-\log n}$
$a_{3,0}$	$a_{3,-1}$	…	$a_{3,-\log n}$
…	…	…	…
$a_{t,0}$	$a_{t,-1}$	…	$a_{t,-\log n}$

b

The a_i's in binary: each $a_{i,j}$ is either σ_i or 0

- degree($Dec^*_{c^*}(\sigma)$) $\leq n$
  ➔ degree($M_{c_1^* c_2^*}(\sigma)$) $\leq 2n$
- Our scheme can do this!!!

Putting Things Together

Bar-Ilan University
Dept. of Computer Science

- Add to public key $d_1, d_2, \ldots, d_t \in [0,2)$
  - $\exists$ sparse S for which $\Sigma_{i \in S}\, d_i = 1/p \bmod 2$
- New secret key is $(\sigma_1, \ldots, \sigma_t)$, char. vector of S
- Also add to public key $u_i = Enc(\sigma_i)$, $i=1,2,\ldots,t$
- Hopefully, scheme remains secure
  - Security with d_i's relies on hardness of "sparse subset sum"
    - Same arguments of hardness as for the approximate-GCD problem
  - Security with u_i's relies on "circular security" (just praying, really)

Computing on Ciphertexts

Tradition of Excellence
Bar-Ilan University
Dept. of Computer Science

- **To "multiply" c_1, c_2** (both with noise $< p/2n$)
  - Evaluate $M_{c_1,c_2}(*)$ on the ciphertexts $u_1, u_2, \ldots, u_t$
  - This is a degree-2n polynomial
  - Result is new c, with noise $<p/2n$
  - Can keep computing on it
- **Same thing for "adding" c_1, c_2**
- **Can evaluate any function**

Ciphertext Distribution

Bar-Ilan University
Dept. of Computer Science

- **May want evaluated ciphertexts to have the same distribution as freshly encrypted ones**
  - Currently they have more noise
- **To do this, make p larger by n bits**
  - "Raw evaluated ciphertext" have noise $< p/2^n$
- **After encryption/evaluation, add noise ~ p/2n**
  - Note: DOES NOT add noise to Enc(σ) in public key
- **Evaluated, fresh ciphertexts now have the same noise**
  - Can show that distributions are statistically close

Conclusions

Bar-Ilan University
Dept. of Computer Science

- Constructed a fully-homomorphic (public key) encryption scheme
- Underlying somewhat-homomorphic scheme relies on hardness of approximate-GCD
- Resulting scheme relies also on hardness of sparse-subset-sum and circular security
- Ciphertext size is ~ n^5 bits
- Public key has ~ n^{10} bits
  - Doesn't quite fit the "efficient" title of the winter school…

More Questions?