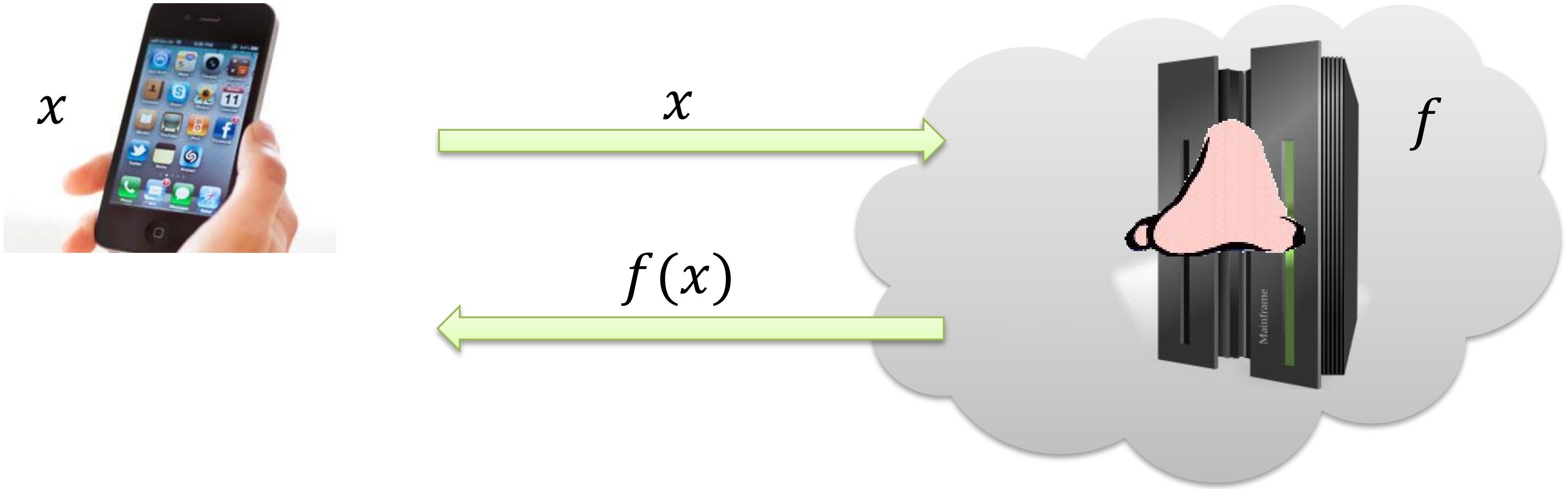
Fully Homomorphic Encryption

Zvika Brakerski

Weizmann Institute of Science

ASCrypto, October 2013

Outsourcing Computation



Email, web-search, navigation, social networking…

Search query, location, business information, medical information…

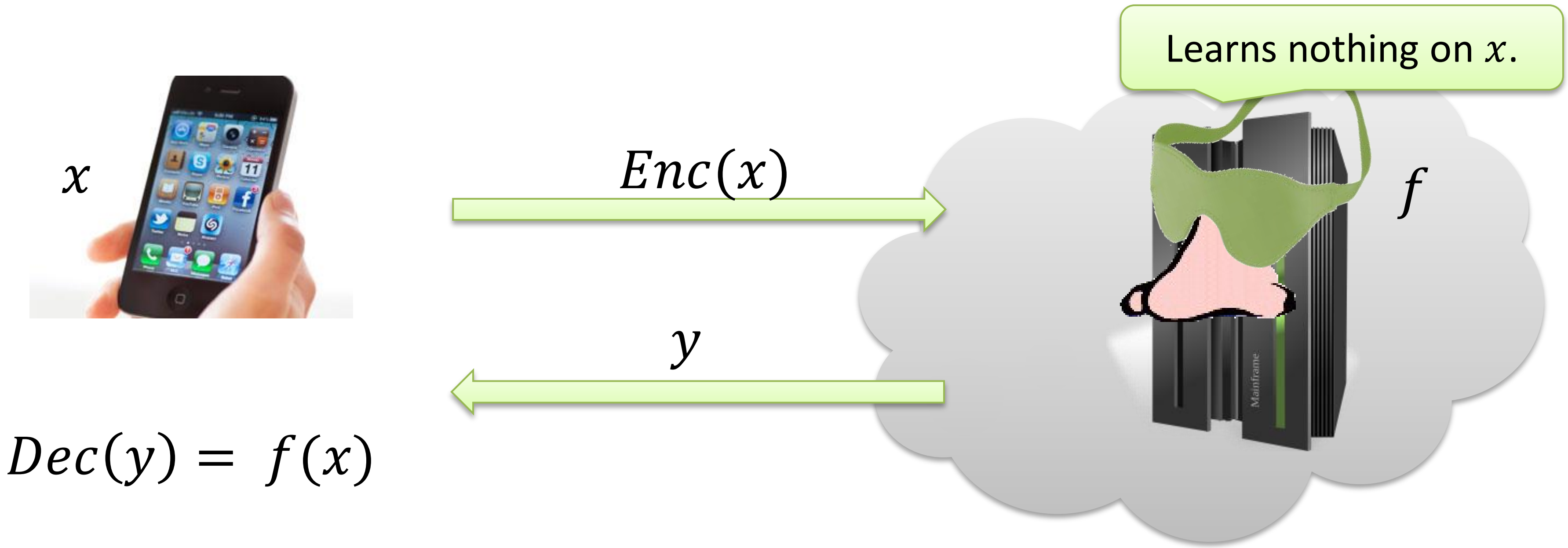
What if x is private?

The Situation Today

We promise we wont look at your data. Honest!

We want real protection.

Outsourcing Computation – Privately

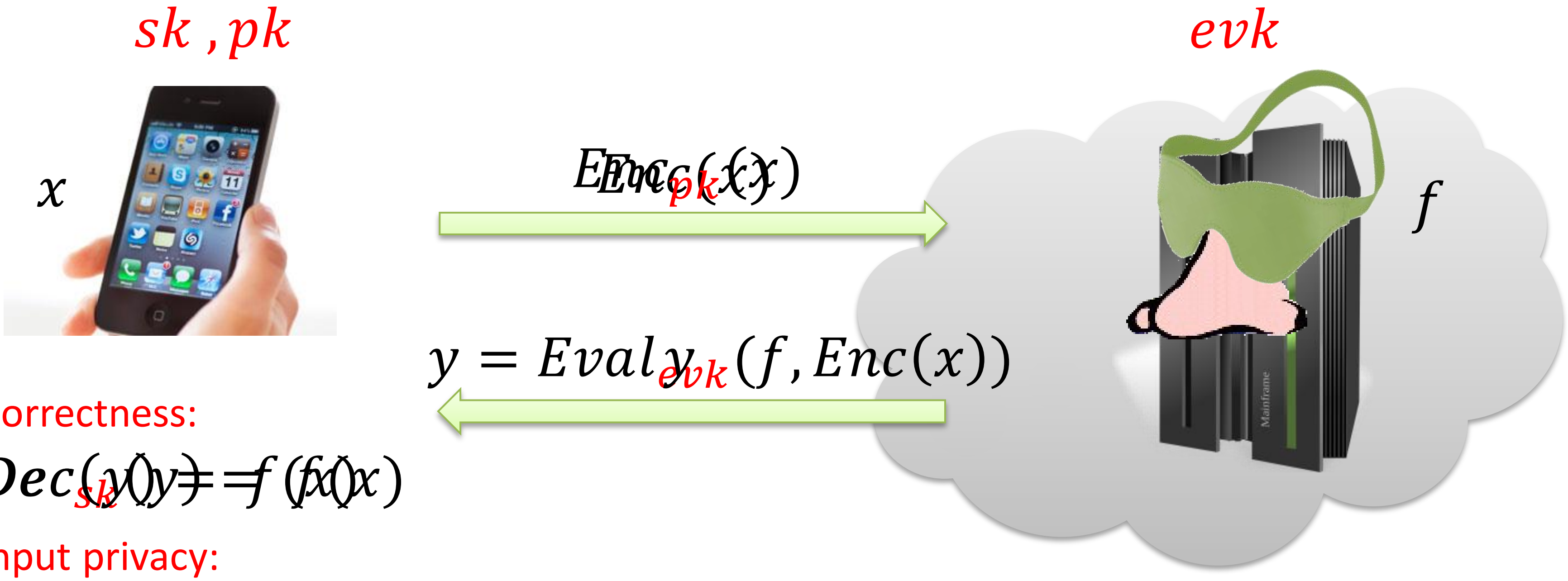


WANTED

Homomorphic Evaluation function:

$Eval\colon f, Enc(x) \rightarrow Enc(f(x))$

Fully Homomorphic Encryption (FHE)



Correctness:

$Dec_{sk}(y) = f(x)$

Input privacy:

$Enc(x) \cong Enc(0)$

Fully Homomorphic = Correctness for any efficient f

= Correctness for universal set

- NAND.
- $(+,\times)$ over $\mathbb{Z}_2$ (= binary XOR, AND)

Trivial FHE?

NOT what we were looking for…

All work is relayed to receiver.

PKE ⇒ "FHE":

- ~~$Keygen$ and Enc: Same as PKE.~~
- ~~$Eval^{FHE}(f,c)$~~
- ~~$Dec^{FHE}_{sk}(f,c) \triangleq f(Dec_{sk}(\underbrace{c}_{Enc\,(x)}))$~~ $= f\Big(Dec_{sk}\big(Enc(x)\big)\Big) = f(x)$

Compact FHE: Dec time does not depend on ciphertext.

⇒ ciphertext length is globally bounded.

In this talk (and in literature) FHE $\triangleq$ Compact-FHE

Trivial FHE?

PKE ⇒ "FHE":

- ~~$Keygen$ and Enc: Same as PKE.~~
- ~~$Eval^{FHE}$ [illegible] (f,c)~~
- ~~$Dec_{sk}^{FHE}(f,c) \triangleq f(Dec_{sk}(c))$~~

This "scheme" also completely reveals f to the receiver.

Can be a problem.

Circuit Privacy: Receiver learns nothing about f (except output).

Compactness ⇒ Circuit Privacy (by complicated reduction) [GHV10]

Circuit private FHE is not trivial to achieve – even non-compact.

In this talk: Only care about compactness, no more circuit privacy.

Applications

In the cloud:

- Private outsourcing of computation.
- Near-optimal private outsourcing of storage (single-server PIR). [G09,BV11b]
- Verifiable outsourcing (delegation). [GGP11,CKV11]
- Private machine learning in the cloud. [GLN12,HW13]

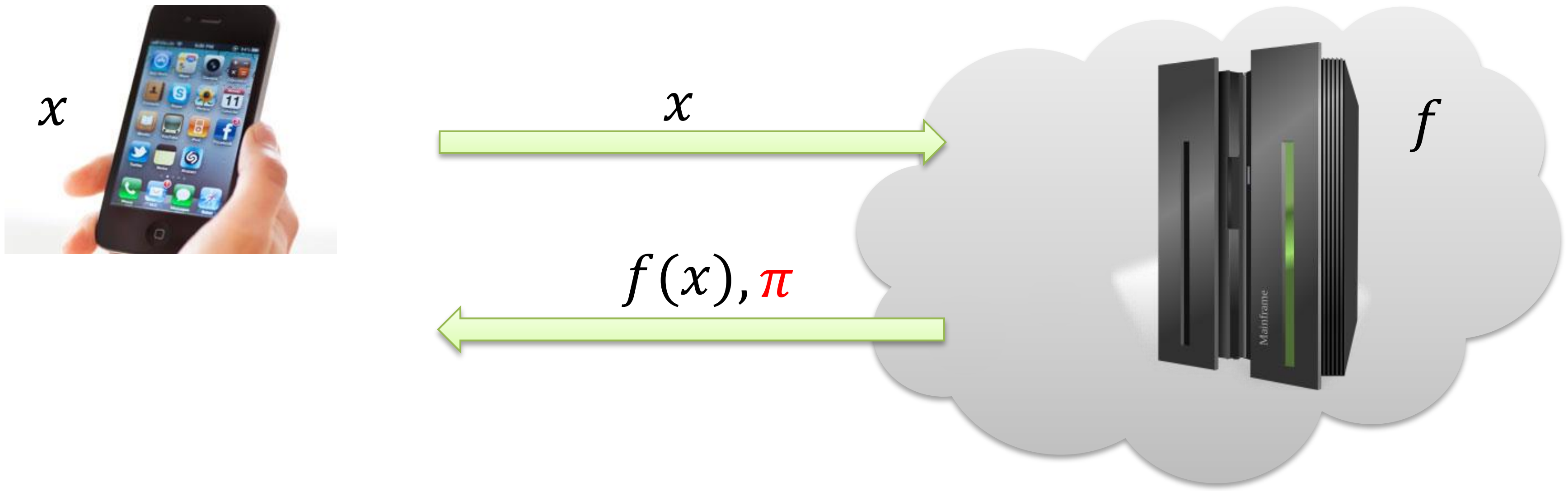
Secure multiparty computation:

- Low-communication multiparty computation. [AJLTVW12,LTV12]
- More efficient MPC. [BDOZ11,DPSZ12,DKLPSS12]

Primitives:

- Succinct argument systems. [GLR11,DFH11,BCCT11,BC12,BCCT12,BCGT13,…]
- General functional encryption. [GKPVZ12]
- Indistinguishability obfuscation for all circuits. [GGHRSW13]

Verifiable Outsourcing (Delegation)



What if the server is cheating?

Can send wrong value of $f(x)$.

Need proof!

FHE $\Rightarrow$ Verifiable Outsourcing

FHE $\Rightarrow$ Verifiability and Privacy.

1. Verifiability with preprocessing under "standard" assumptions: [GGP10, CKV10].

2. Less standard assumptions but without preprocessing via SNARGs/SNARKs [DCL08,BCCT11,…] (uses FHE or PIR).

Pre-FHE solutions: multiple rounds [K92] or random oracles [M94].

FHE ⇒ Verifiable V10]

Preprocessing:

$c_0 = Enc(0)$
$z_0 = Eval(f, c_0)$

But preprocessing is as hard as computation!

sk, pk

x

$c_x = Enc(x), c_0$

y_x, y_0

evk

Verification:

Check $y_0 = z_0$?

Yes ⇒ output $Dec(y_x)$
No ⇒ output $\perp$

Server executes
$y = Eval(f, c)$

Idea: "Cut and choose"

c_x, c_0 look the same ⇒ cheating server will be caught w.p. ½
(easily amplifiable)

FHE ⇒ Verifiable Outsourcing [CKV10]

Preprocessing:

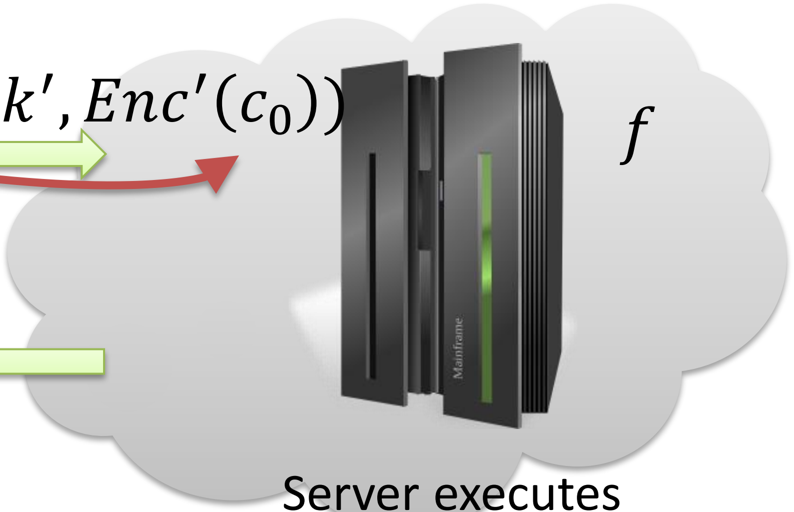
$c_0 = Enc(0)$
$z_0 = Eval(f, c_0)$

$sk\ , pk$

x

evk

f



$(evk'', Enc''(c_x)), (evk', Enc'(c_0))$

y''_x, y'_0

Verification:

Check $Dec'(y'_0) = z_0$?

Yes ⇒ output $Dec''(Dec(y_x))$
No ⇒ output $\perp$

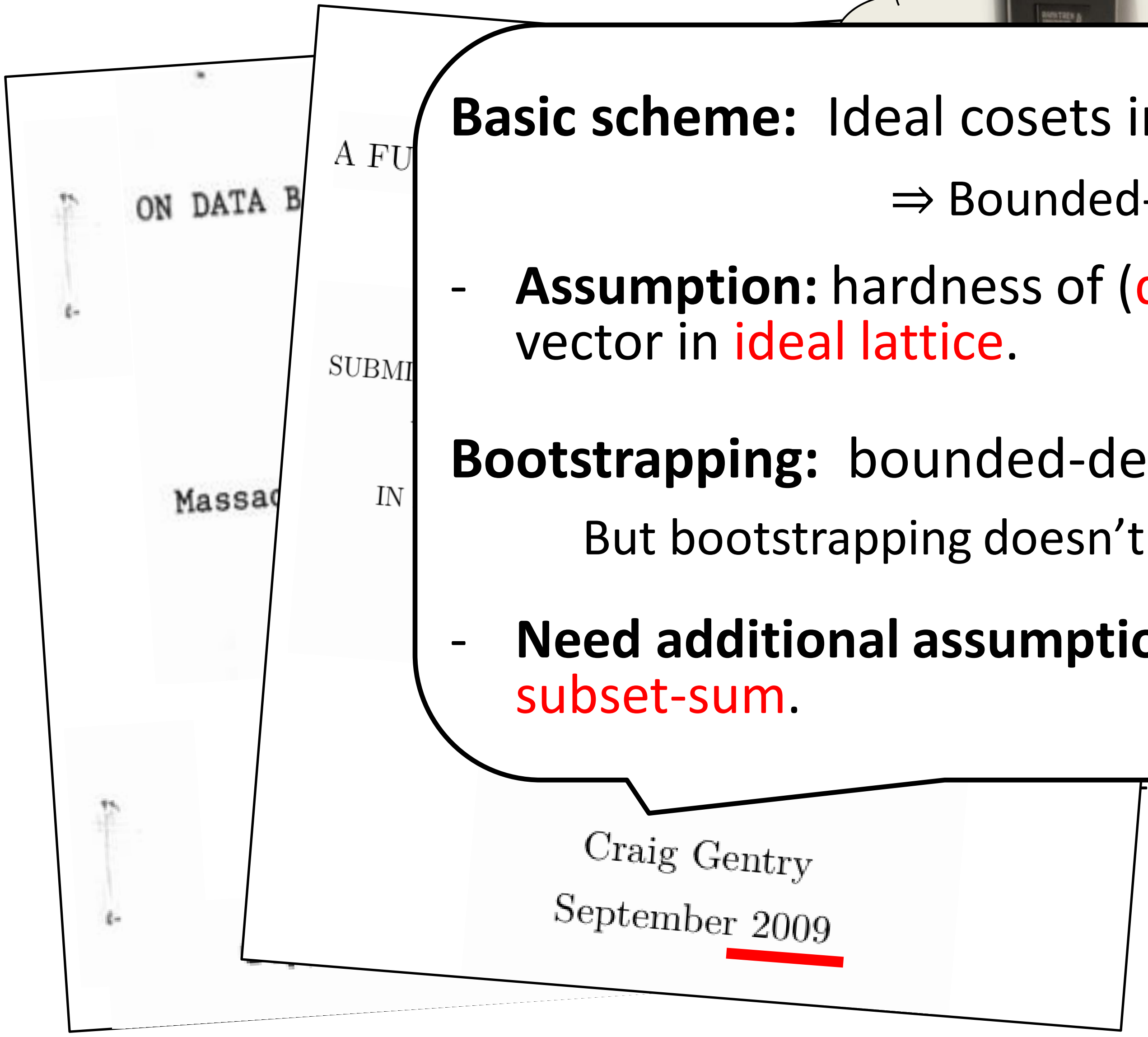
Server executes
$y' = Eval'(Eval(f,\cdot), c')$
$y'' = Eval''(Eval(f,\cdot), c'')$

Server is not allowed to know if we accept/reject!

Idea: Outer layer keeps server "c

⇒ Can recycle z_0 for future computations.

FHE Timeli



Basic scheme: Ideal cosets in polynomial rings.

⇒ Bounded-depth homomorphism.

- **Assumption:** hardness of (quantum) apx. short vector in ideal lattice.

Bootstrapping: bounded-depth HE ⇒ full HE.

But bootstrapping doesn't apply to basic scheme...

- **Need additional assumption:** hardness of sparse subset-sum.

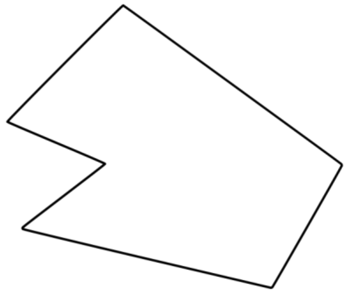
… is it even possible?

The FHE Challenge

Make it simpler.

Simplified basic scheme [vDGHV10,BV11a]

- Under similar assumptions.



Make it more secure.

?

Make it practical.

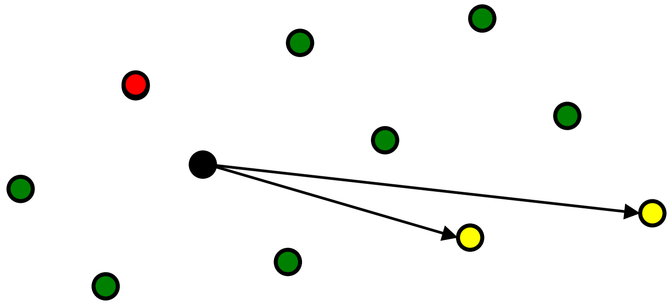
Optimizations [SV10,SS10,GH10]

FHE without Ideals [BV11b]

Linear algebra instead of polynomial rings

Assumption: Apx. short vector in **arbitrary** lattices (via LWE).

Shortest-vector Problem (SVP):



Fundamental algorithmic problem –
extensively studied.

[LLL82,K86,A97,M98,AKS03,MR04,MV10]

FHE without Ideals [BV11b]

Linear algebra instead of polynomial rings

Assumption: Apx. short vector in **arbitrary** lattices (via LWE).

- **Basic scheme:** noisy linear equations over $\mathbb{Z}_q$.
  - Ciphertext is a linear function $c(x)$ s.t. $c(sk) \approx m$.
  - Add/multiply functions for homomorphism.
  - Multiplication raises degree $\Rightarrow$ use **relinearization**.
- **Bootstrapping:** Use **dimension-modulus reduction** to shrink ciphertexts.

- Simpler: straightforward presentation.
- More secure: based on a standard assumption.
- Efficiency improvements.

Concurrently [GH11]: Ideal lattice based scheme without squashing.

FHE without Ideals

Follow-ups:

- [BGV12]: Improved parameters.
  - Even better security.
  - Improved efficiency in ring setting using “batching”.
  - Batching without ideals in [BGH13].
- [B12]: Improved security.
  - Security based on classical lattice assumptions.
  - Explained in blog post [BB12].

Various optimizations, applications and implementations:

[LNV11, GHS12a, GHS12b, GHS12c, GHPS12, AJLTVW12, LTV12, DSPZ12, FV12, GLN12, BGHWW12,HW13 …]

The “Approximate Eigenvector” Method [GSW13]

Ciphertexts = Matrix

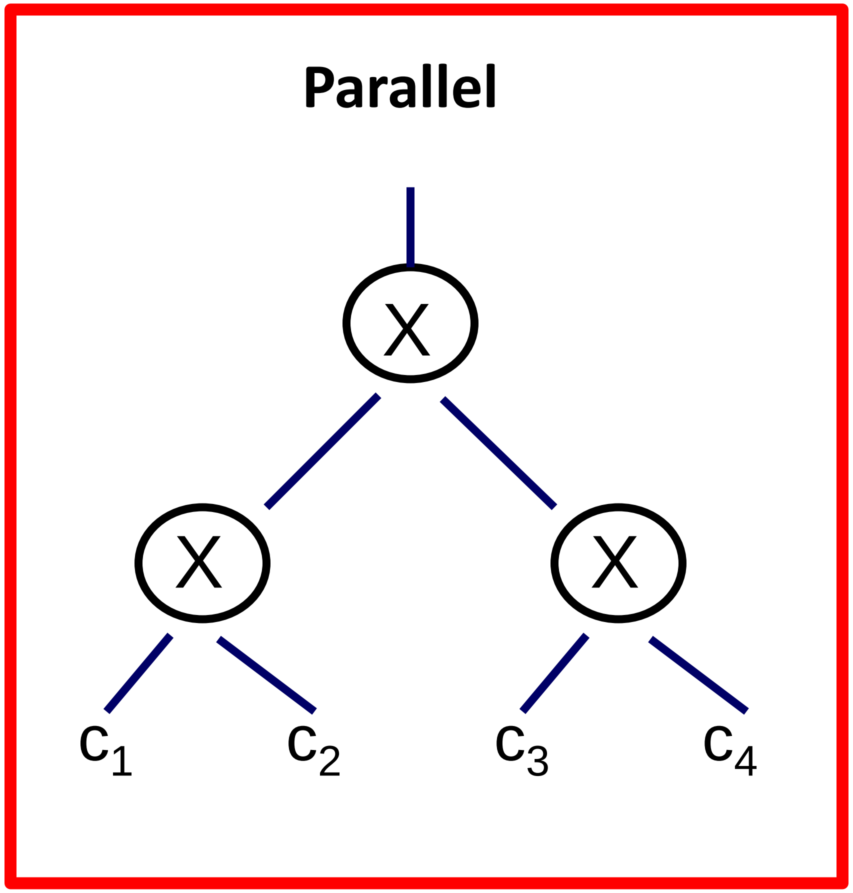
Same assumption and keys as before – ciphertexts are different

- **Basic scheme:** Approximate eigenvector over $\mathbb{Z}_q$.
  - Ciphertext is a matrix C s.t. $C \cdot sk \approx m \cdot sk$.
  - Add/multiply matrices for homomorphism*.
- **Bootstrapping:** Same as previous schemes.

- Simpler: straightforward presentation.
- New and exciting applications “for free”! IB-FHE, AB-FHE.
- Same security as [BGV12, B12].
- Unclear about efficiency: some advantages, some drawbacks.

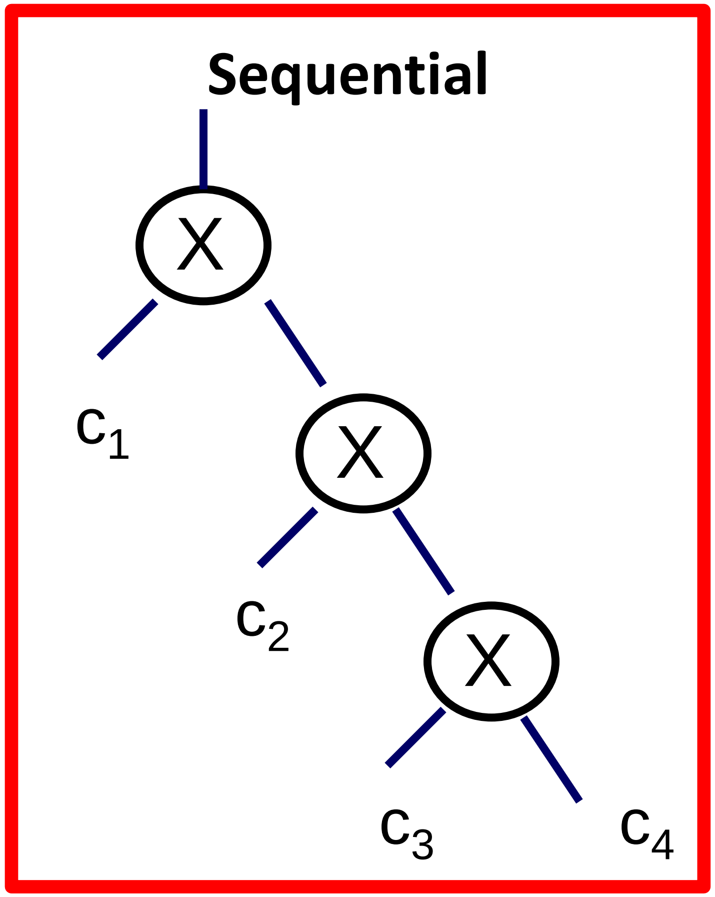
Sequentialization [BV13]

What is the best way to evaluate a product of k numbers?



Conventional wisdom

vs.



Actually better
(if done right)

Sequentialization [BV13]

Barrington's Theorem [B86]: Every depth d computation can be transformed into a width-5 depth 4^d **branching program**.

A sequential model of computation

- Better security – breaks barrier of [BGV12, B12,GSW13].
- Using dimension-modulus reduction (from [BV11b]) $\Rightarrow$ same hardness assumption as non homomorphic encryption.
- Short ciphertexts.

Efficiency

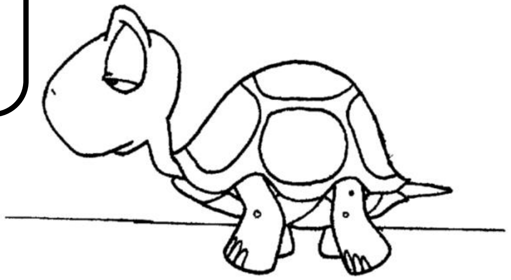
See also HElib

`https://github.com/shaih/HElib`

Star

Implementations of [BGV12] by [GHS12c,CCKLLTY13]≈5 min/input

2-years ago it was 3 min/**gate** [GH10]



Limiting factors:

- Circuit representation.
- Bootstrapping.
- Key size.

New works [GSW13,BV13] address some of these issues, but have other drawbacks

⇒ To be practical, we need to improve the theory.

Hybrid FHE

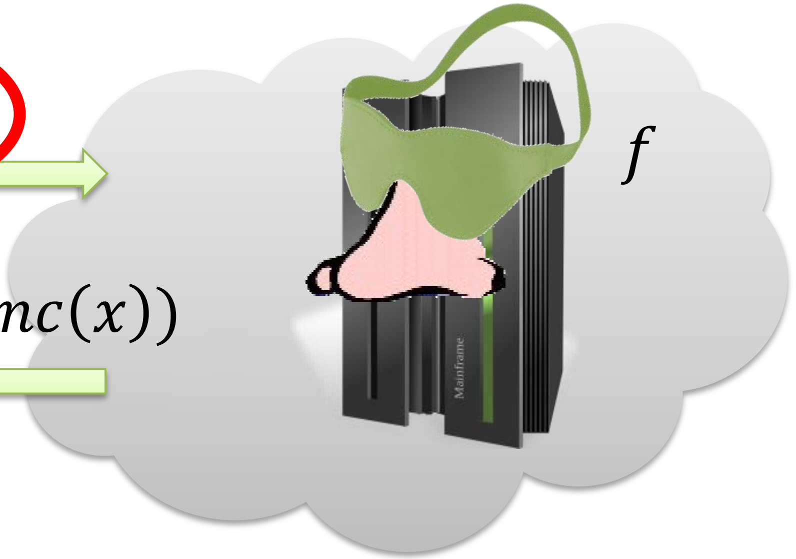
$sk\ , pk$

evk

x

$Enc_{pk}(x)$

f



$y = Eval_{evk}(f, Enc(x))$

$Dec_{sk}(y) = f(x)$

- In known FHE encryption is slow and ciphertexts are long.
- In **symmetric** encryption (e.g. AES) these are better.

Best of both worlds?

Hybrid FHE

sym

$sk\ , pk$

x

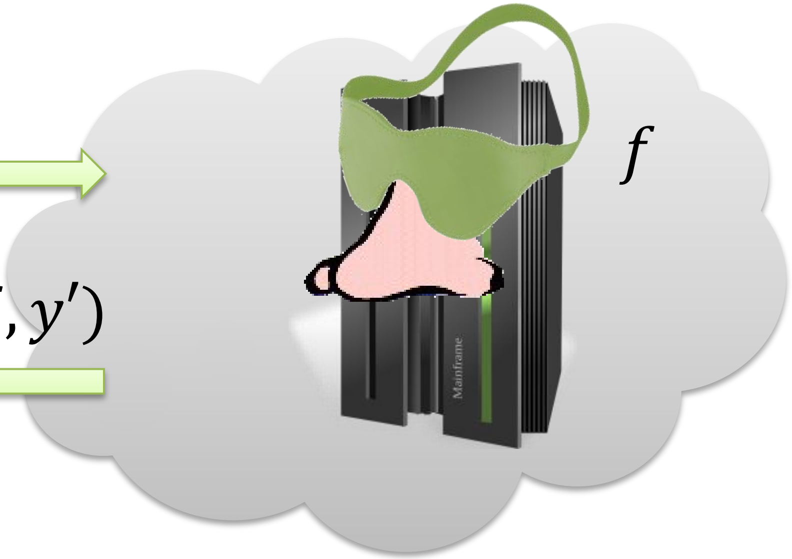
c=$Enc_{sym}(x)$

$y = Eval_{evk}(f, y')$

$Dec_{sk}(y) = \ f(x)$

$Enc_{pk}(sym)$

evk



f

Easy to encrypt, ciphertext is short… But how to do Eval?

Define: $h(z) = SYM\_Dec_z(c)$

Server Computes: $y' = Eval_{evk}(h, Enc_{pk}(sym))$

$$\Rightarrow\ y' = Enc\big(h(sym)\big) = Enc\left(SYM\_Dec_{sym}(c)\right) = Enc_{pk}(x)$$

Approximate Eigenvector Method [GSW13]

Observation: Let C_1, C_2 be matrices with the same eigenvector $\vec{s}$, and let m_1, m_2 be their respective eigenvalues w.r.t $\vec{s}$. Then:

1. $C_1 + C_2$ has eigenvalue $(m_1 + m_2)$ w.r.t $\vec{s}$.
2. $C_1 \cdot C_2$ (and also $C_2 \cdot C_1$) has eigenvalue $m_1 m_2$ w.r.t $\vec{s}$

Say over $\mathbb{Z}_q$

Idea: $\vec{s}$ = secret key, C = ciphertext, and m = message.

$\Rightarrow$ Homomorphism for addition and multiplication.

$\Rightarrow$ Full homomorphism!

Insecure! Eigenvectors are easy to find.

What about **approximate** eigenvectors?

Approximate Eigenvector Method [GSW13]

$$\boxed{C \cdot \vec{s} = m\vec{s} + \vec{e} \approx m\vec{s}}$$

How to decrypt? Must have restriction on $\|\vec{e}\|$

Suppose $\vec{s}[1] = q/2$, and $m \in \{0,1\}$

$\Rightarrow (C \cdot \vec{s})[1] = \frac{q}{2} m + \vec{e}[1]$ Find m by rounding

Condition for correct decryption: $\|\vec{e}\| < q/4$.

Approximate Eigenvector Method [GSW13]

$$C_1 \cdot \vec{s} = m_1\vec{s} + \vec{e}_1 \qquad C_2 \cdot \vec{s} = m_2\vec{s} + \vec{e}_2$$

$$\|\vec{e}_1\| \ll q \qquad \|\vec{e}_2\| \ll q$$

Goal: $C_1, C_2 \Rightarrow \quad C_{add} = Enc(m_1 + m_2)\,, C_{mult} = Enc(m_1 m_2).$

$C_{add} = C_1 + C_2$:

$$\begin{aligned}(C_1+C_2) \cdot \vec{s} &= C_1\vec{s} + C_2\vec{s} \\ &= m_1\vec{s} + \vec{e}_1 + m_2\vec{s} + \vec{e}_2 \\ &= (m_1+m_2)\vec{s} + \underbrace{(\vec{e}_1+\vec{e}_2)}_{\vec{e}_{add}}\end{aligned}$$

Noise grows a little

Approximate Eigenvector Method [GSW13]

$C_1 \cdot \vec{s} = m_1\vec{s} + \vec{e}_1$ $\qquad$ $C_2 \cdot \vec{s} = m_2\vec{s} + \vec{e}_2$

$\|\vec{e}_1\| \ll q$ $\qquad$ $\|\vec{e}_2\| \ll q$

Goal: $C_1, C_2 \Rightarrow \quad C_{add} = Enc(m_1 + m_2)\,,\, C_{mult} = Enc(m_1 m_2)$.

$C_{mult} = C_1 \cdot C_2$:

Can also use $C_2 \cdot C_1$

$$\begin{aligned}(C_1 \cdot C_2) \cdot \vec{s} &= C_1(m_2\vec{s} + \vec{e}_2)\\ &= m_2 C_1\vec{s} + C_1\vec{e}_2\\ &= m_2(m_1\vec{s} + \vec{e}_1) + C_1\vec{e}_2\\ &= m_2 m_1\vec{s} + \underbrace{m_2\vec{e}_1 + C_1\vec{e}_2}_{\vec{e}_{mult}}\end{aligned}$$

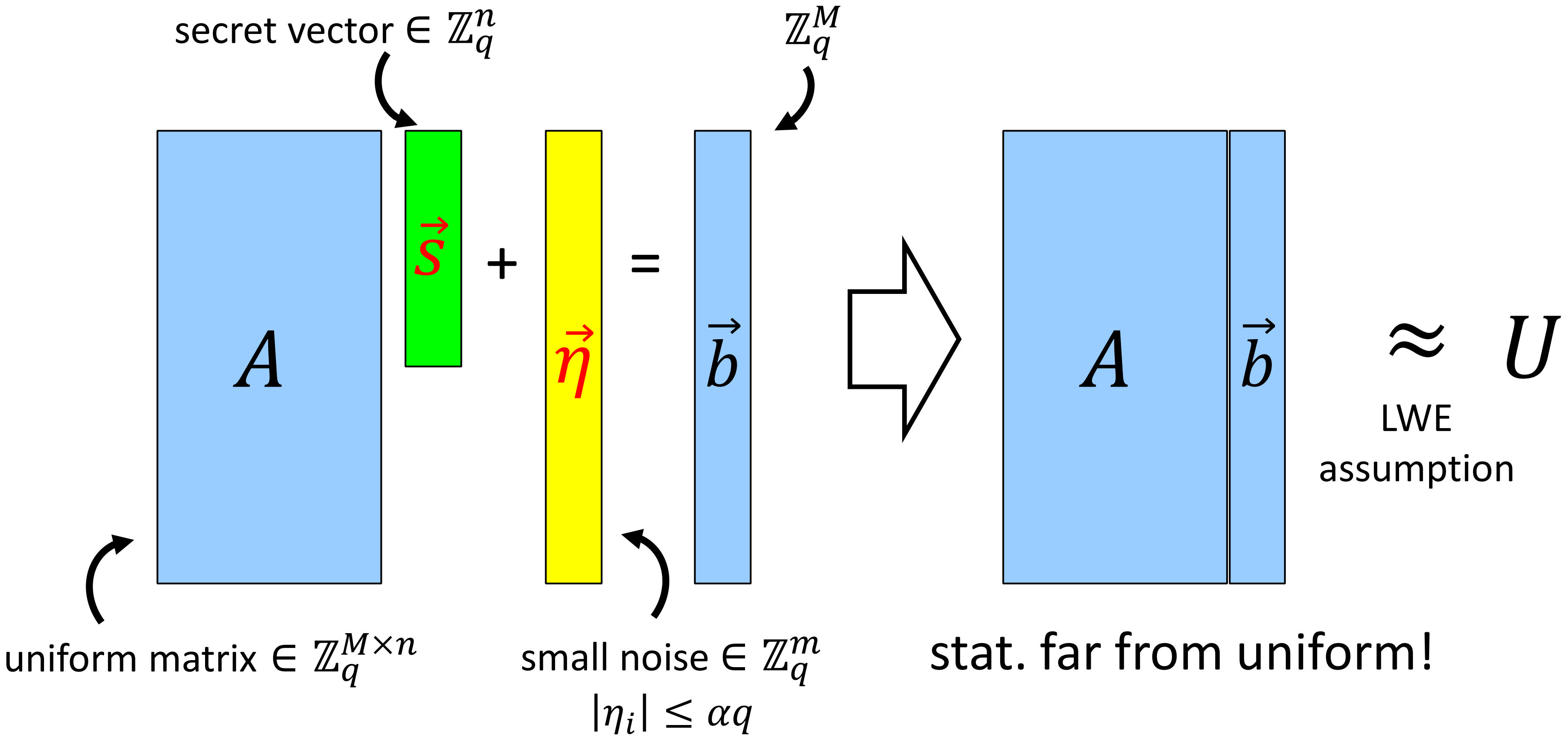
Noise grows.
But by how much?

Plan for Technical Part

1. Constructing approximate eigenvector scheme.
2. Sequentialization.
3. Bootstrapping.
4. Open problems and limits on FHE.

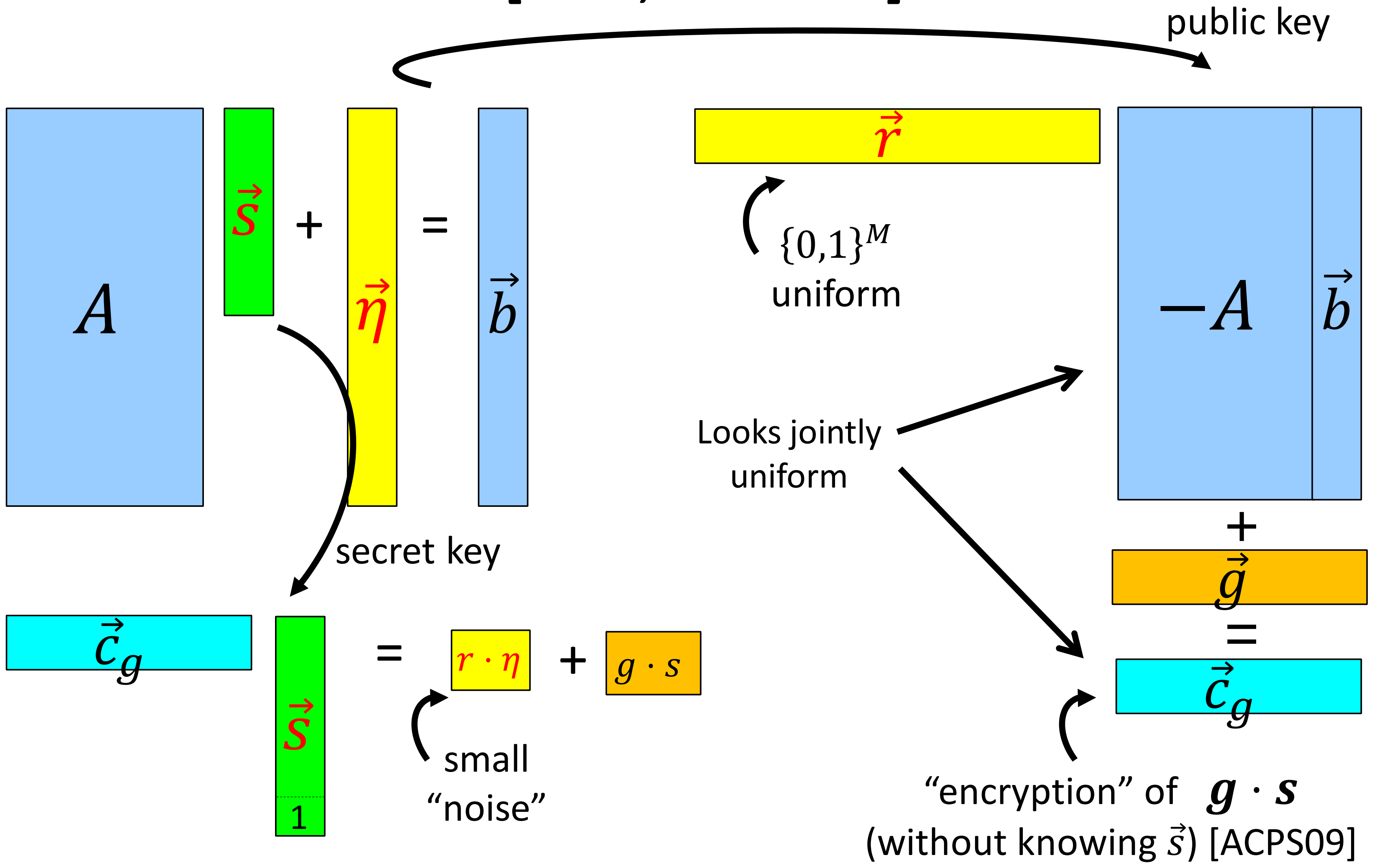
Learning with Errors (LWE) [R05]

Random noisy linear equations ≈ uniform

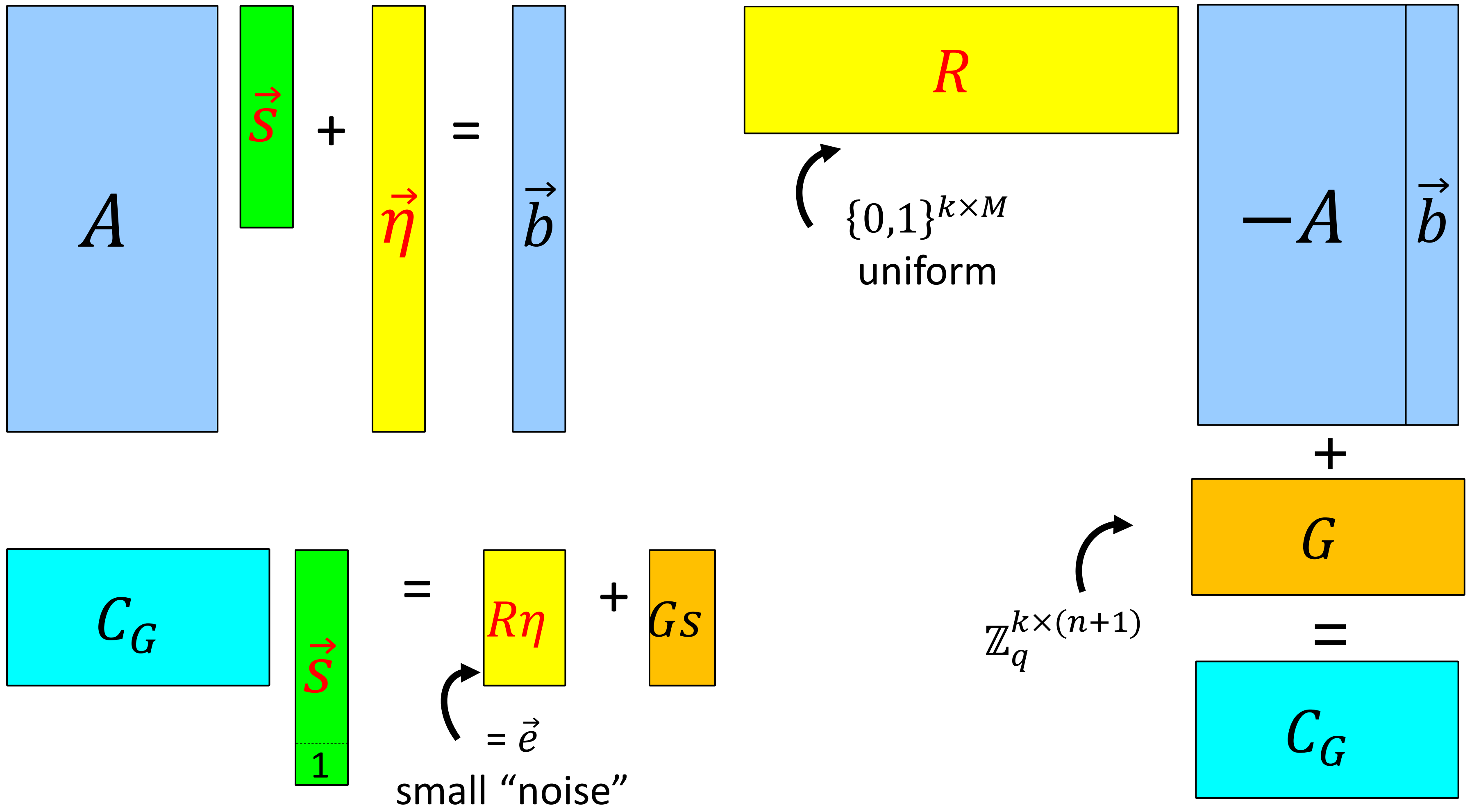


As hard as (n/α)-apx. short vector in **worst case** n-dim. lattices [R05, P09]

Encryption Scheme from LWE [R05,ACPS09]



Encryption Scheme from LWE [R05,ACPS09]



Approx. Eigenvector Encryption

Goal: Encrypt message $m \in \{0,1\}$

Idea: $Enc(m) = C_{m \cdot I}$

$$\Rightarrow \; C_{m \cdot I} \cdot \vec{s} = \vec{e} + mI\vec{s} = m \cdot \vec{s} + \vec{e}$$

As we saw:

$$\begin{aligned} C_1 \cdot C_2 \cdot \vec{s} &= C_1 \cdot (\vec{e}_2 + m_2\vec{s}) \\ &= C_1 \cdot \vec{e}_2 + m_2 \cdot C_1 \cdot \vec{s} \\ &= \underbrace{C_1 \cdot \vec{e}_2}_{\text{HUGE noise}} + \underbrace{m_2\vec{e}_1}_{\text{small noise}} + \underbrace{m_1 m_2 \vec{s}}_{\text{desired output}} \end{aligned}$$

Need to reduce the norm of C_1

Solution: binary decomposition

Binary Decomposition

Break each entry in C to its binary representation

$$C = \begin{bmatrix} 3 & 5 \\ 1 & 4 \end{bmatrix} \ (mod\ 8) \Rightarrow bits(C) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \ (mod\ 8)$$

Small entries like we wanted!

But product with $\vec{s}$ now meaningless

Consider the "reverse" operation:

$$bits(C) \cdot \underbrace{\begin{bmatrix} 4 & 0 \\ 2 & 0 \\ 1 & 0 \\ 0 & 4 \\ 0 & 2 \\ 0 & 1 \end{bmatrix}}_{G} = C \quad \Rightarrow$$

$$C \cdot \vec{s} = bits(C) \cdot G \cdot \vec{s} = bits(C) \cdot \vec{s}^*$$

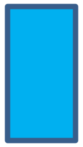
$$\vec{s}^* = G \cdot \vec{s}$$

"powers of 2" vector

Contains $q/2$ as an element

Approx. Eigenvector Encryption

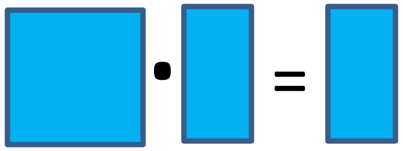
$$Enc(m) = C_{m\cdot G} \in \mathbb{Z}_q^{\overbrace{((n+1)\log q)}^{N}\times(n+1)}$$



$$C_{nand} = G - bits(C_1)\cdot C_2$$

$$\Rightarrow\ C_{m\cdot G}\cdot\vec{s} = \vec{e} + m\cdot G\cdot\vec{s}$$

$$\boxed{C_{mult} = bits(C_1)\cdot C_2}$$

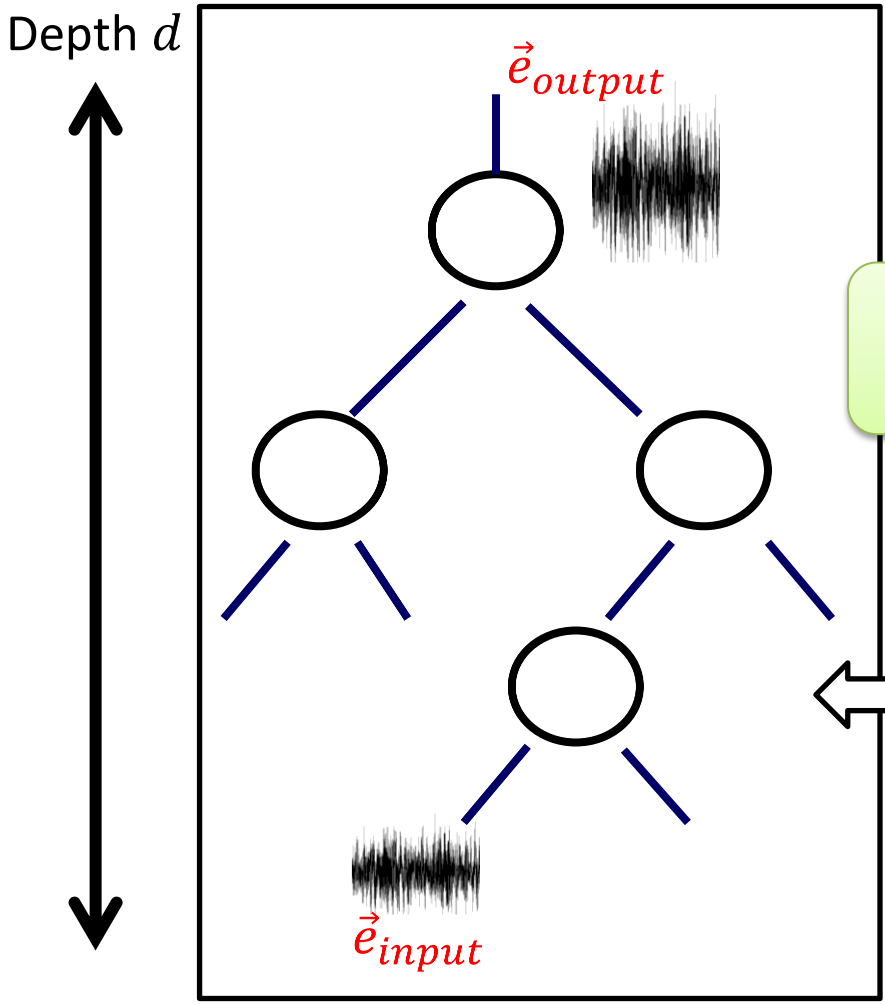


$$\begin{aligned}
bits(C_1)\cdot C_2\cdot\vec{s} &= bits(C_1)\cdot(\vec{e}_2 + m_2 G\vec{s})\\
&= bits\,(C_1)\cdot\vec{e}_2 + m_2\cdot bits(C_1)\cdot G\cdot\vec{s}\\
&= bits\,(C_1)\cdot\vec{e}_2 + m_2\cdot C_1\cdot\vec{s}\\
&= \underbrace{bits\,(C_1)\cdot\vec{e}_2}_{\text{small-ish}} + \underbrace{m_2\cdot\vec{e}_1}_{\text{small}} + \underbrace{m_1\cdot m_2\cdot G\cdot\vec{s}}_{\text{desired output}}
\end{aligned}$$

$$\boxed{\|\vec{e}_{nand}\| \le N\cdot\|\vec{e}_2\| + m_2\cdot\|\vec{e}_1\| \le (N+1)\cdot\max\{\|\vec{e}_1\|,\|\vec{e}_2\|\}}$$

Homomorphic Circuit Evaluation

Noise grows during homomorphic evaluation



$$\|\vec{e}_{output}\| \leq (N+1)^d \cdot M\alpha q$$
$$\approx N^d \alpha q$$

⇒ Decryption succeeds if $\alpha \ll 1/N^d$.

⋮

$$\|\vec{e}_{i+1}\| \leq (N+1)\|\vec{e}_i\|$$

$$\|\vec{e}_{input}\| \leq M\alpha q$$

Full Homomorphism

$$\boxed{\begin{array}{c}\alpha \le N^{-d} \\ d_{hom} \approx \log(1/\alpha)\end{array}}$$

1. If depth upper-bound is known ahead of tim

Set $N \ge d^2$; $\alpha = 2^{-\sqrt{N}} \Rightarrow \log(1/\alpha) = d$

Leveled FHE: Parameters (evk) grow with d.

Undesirable:

- Huge parameters.
- Low security.
- Inflexible.

2. Single scheme for any poly depth.

Bootstrap!

The Bootstrapping Theorem

(Proof to come)

Additional condition, to be discussed.

Homomorphic ⇒ fully homomorphic

when $d_{dec} < d_{hom}$

- d_{dec} = depth of the decryption circuit.
- d_{hom} = maximal homomorphic depth.

In our scheme: $d_{dec} = \log N \;\; \Rightarrow$ FHE if $\alpha < N^{-\log N}$

Quasi-polynomial approximation for short vector problems (same factor as [BGV12,B12])

Non-homomorphic schemes only need $N^{O(1)}$ approximation

A Taste of Sequentialization [BV13]

$$\vec{e}_{mult} = bits\,(C_1) \cdot \vec{e}_2 + m_2 \cdot \vec{e}_1$$

Asymmetric!

Important observations:

1. $\vec{e}_1$ gets multiplied by 0/1 ; $\vec{e}_2$ can get multiplied by N.

2. $m_2 = 0 \;\Rightarrow\; \vec{e}_1$ has no effect!

Conclusion: The order of multiplication matters.

Want to multiply C_A, C_B s.t. $\vec{e}_A \gg \vec{e}_B$.

Which is better: $bits(C_A) \cdot C_B$ or $bits(C_B) \cdot C_A$?

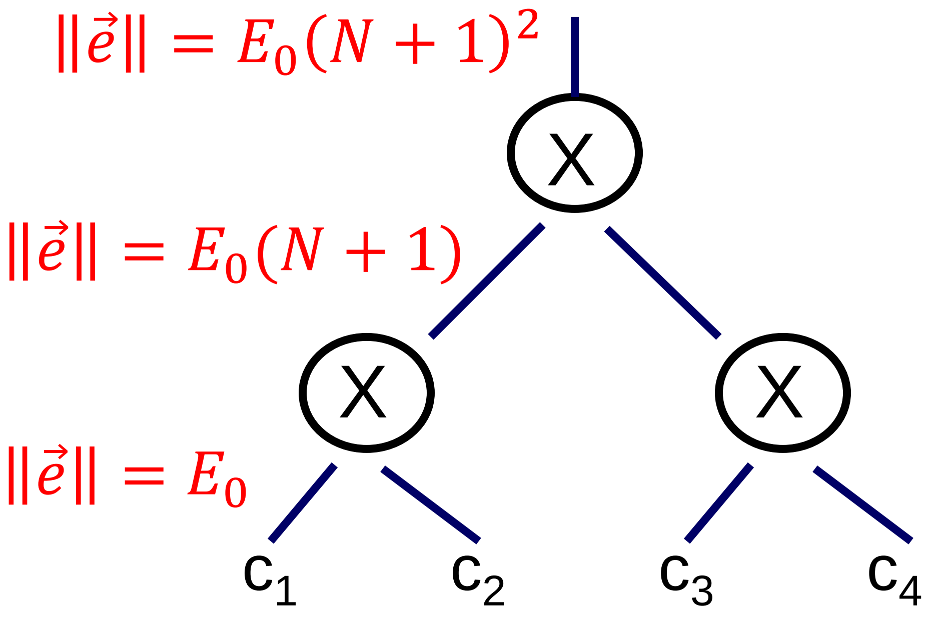
A Taste of Sequentialization [BV13]

$$\vec{e}_{mult} = bits\,(C_1) \cdot \vec{e}_2 + m_2 \cdot \vec{e}_1$$

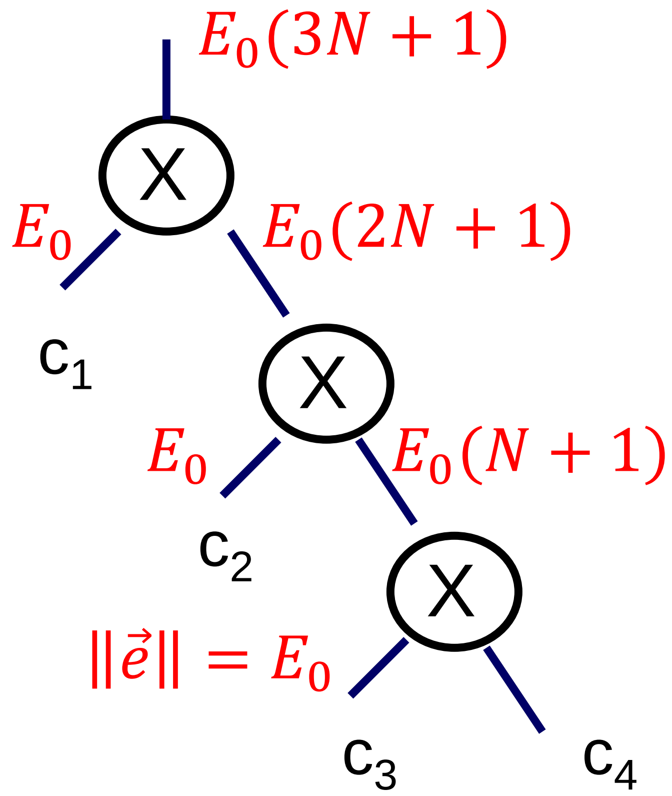
Task: Multiply 4 ciphertexts $C_1, \ldots, C_4$

Winner!

Multiplication Tree



Sequential Multiplier



Bootstrapping

Homomorphic $\Rightarrow$ fully homomorphic when

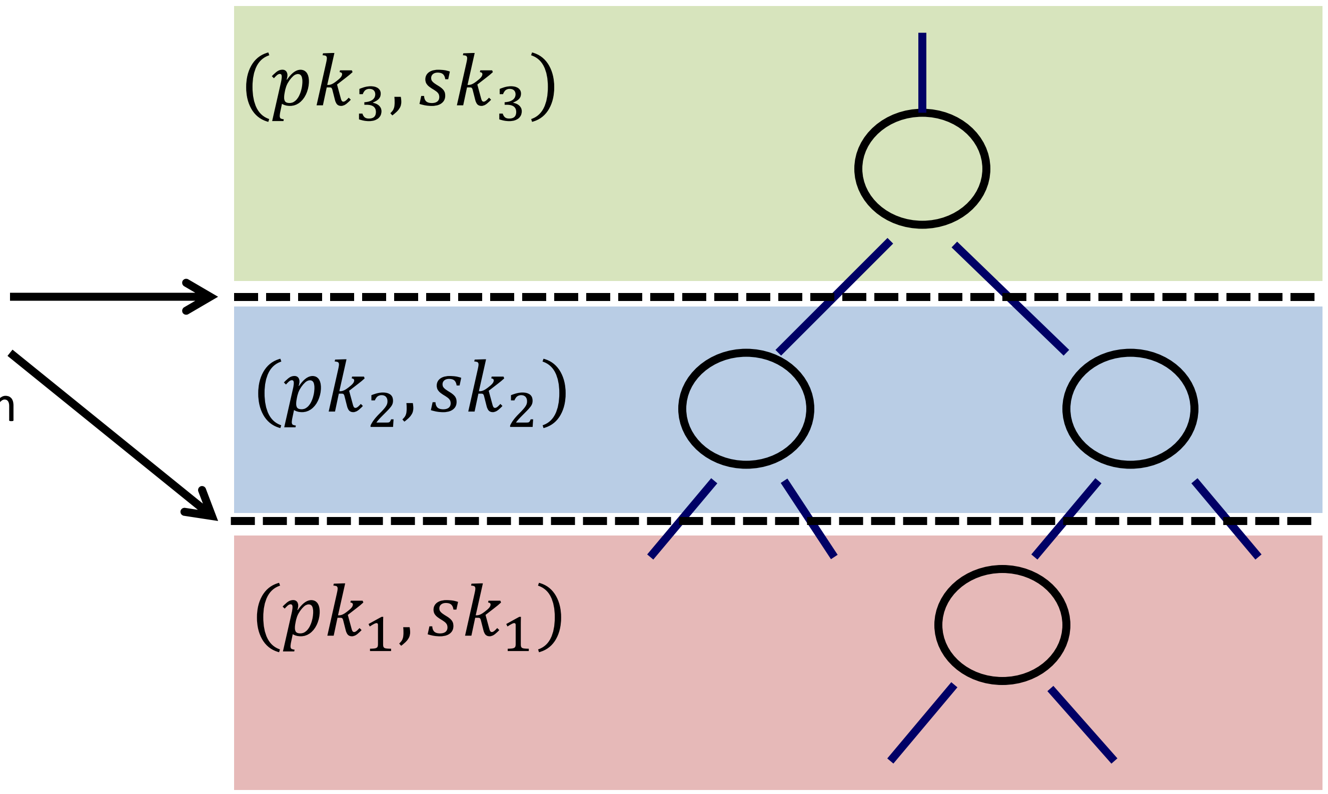
$$d_{dec} < d_{hom}$$

- d_{dec} = depth of the decryption circuit.
- d_{hom} = maximal homomorphic depth.

Bootstrapping

Given scheme with bounded d_{hom}
How to extend its homomorphic capability?

Idea: Do a few operations, then "switch" to a new instance



How to Switch Keys

We have seen this before!

Hybrid FHE

Hybrid FHE

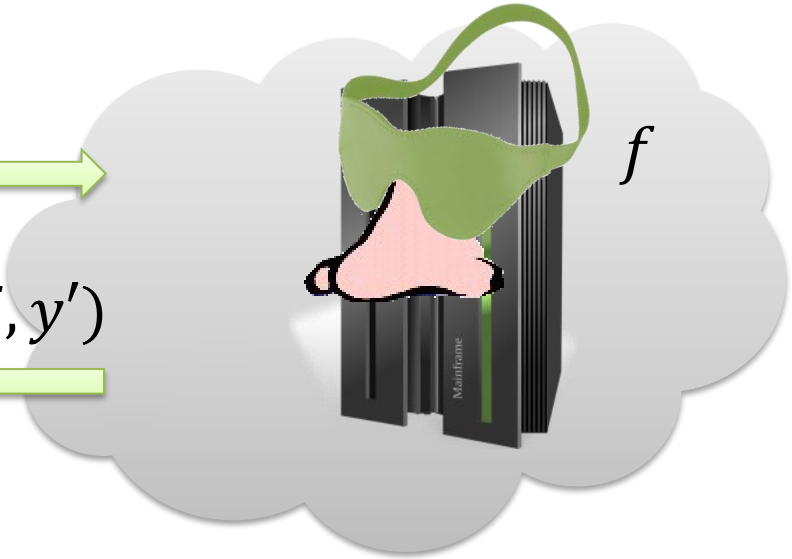
sym

sk, pk

x

$Enc_{pk}(sym)$

evk



f

$$\xrightarrow{c=Enc_{sym}(x)}$$

$$\xleftarrow{y = Eval_{evk}(f, y')}$$

$Dec_{sk}(y) = f(x)$

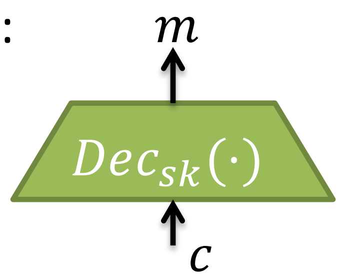
Define: $h(z) = SYM\_Dec_{z}(c)$

Server Computes: $y' = Eval_{evk}(h, Enc_{pk}(sym))$

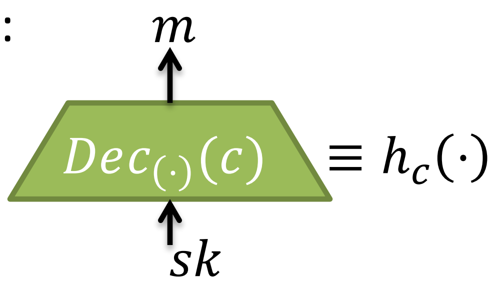
$$\Rightarrow y' = Enc(h(sym)) = Enc\left(SYM\_Dec_{sym}(c)\right) = Enc_{pk}(x)$$

How to Switch Keys

Decryption circuit:



Dual view:



$$h_c(sk) = Dec_{sk}(c) = m$$

Key switching procedure $(sk_1, pk_1) \to (sk_2, pk_2)$:

Input: $c = Enc_{pk_1}(m)$

Server aux info: $aux = Enc_{pk_2}(sk_1)$ (ahead of time)

Output: $Eval_{pk_2}(h_c, aux)$

Eval depth = d_{dec}

$$\begin{aligned} Eval_{pk_2}(h_c, aux) &= Eval_{pk_2}\left(h_c, Enc_{pk_2}(sk_1)\right) \\ &= Enc_{pk_2}(h_c(sk_1)) = Enc_{pk_2}\left(Dec_{sk_1}(c)\right) \\ &= Enc_{pk_2}(m) \end{aligned}$$

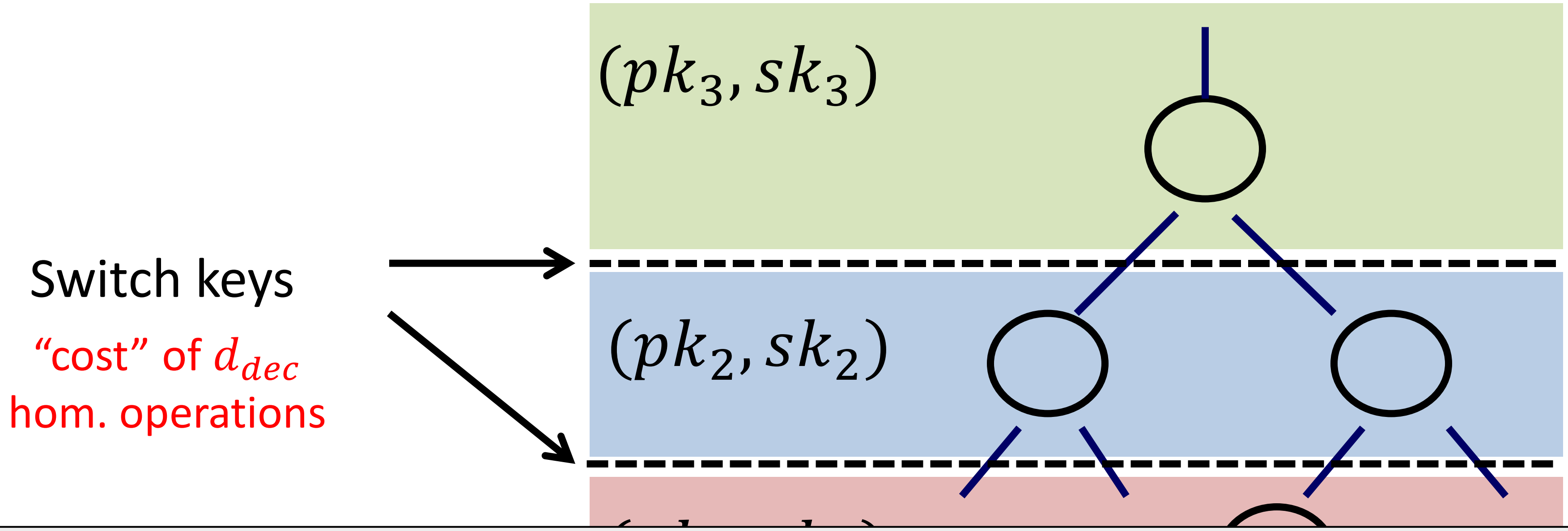
Bootstrapping

Given scheme with bounded

How to extend its homomorphic

Need to generate many keys…

Idea: Do a few operations, then "switch" to a new instance



Conclusion: Bootstrapping if $d_{hom} \geq d_{dec} + 1$

Bootstrapping

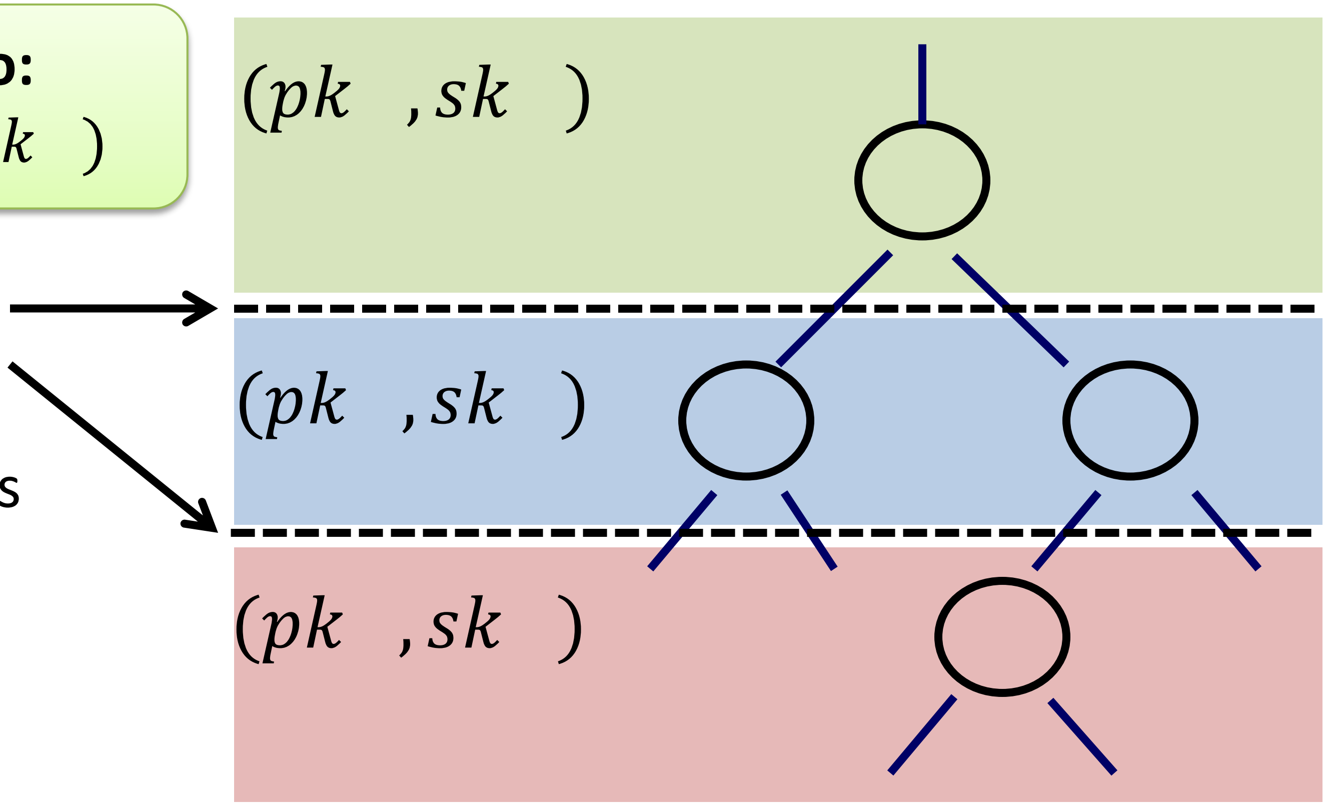
Given scheme with bounded d_{hom}.
How to extend its homomorphic capability?

Idea: Do a few operations, then "switch" ~~to a new instance~~

Server aux info:
$aux = Enc_{pk}\ (sk\ )$

Switch from the key to itself!

Key switching works



Circular Security

Is it secure to publish $aux = Enc_{pk}(sk)$

Intuitively: Yes, encryption hides the message.

Formally: Security does not extend.

What can we do about it?

Option 1: Assume it's secure – no attack is known.

Option 2: Use a sequence of keys.

$\Rightarrow$ No. of keys proportional to computation depth (leveled FHE).

Short keys without circular assumption ?

[BV11a]: Circular secure "somewhat" homomorphic scheme.

Diversity

- Other (older) schemes with similar properties [AD97, GGH97, R03, R05, …] ⇒ homomorphism

  But all are lattice based

- [BL11] FHE from a noisy decoding problem.

  broken

  [B13]: Homomorphicly "clean up" the noise ⇒ break security.

  ⇒ "Too much" homomorphism is a bad sign.

What We Saw Today

- Definition of FHE.
- Applications.
- Historical perspective and background.
- Constructing HE using the approximate eigenvector method.
- Sequentialization.
- Bootstrapping.
- Limits on HE.

Open Problems

- Short keys without circular security.
- FHE from different assumptions.
- CCA1 secure FHE.
- Bounded malleability.
- Improved efficiency.

Thank You