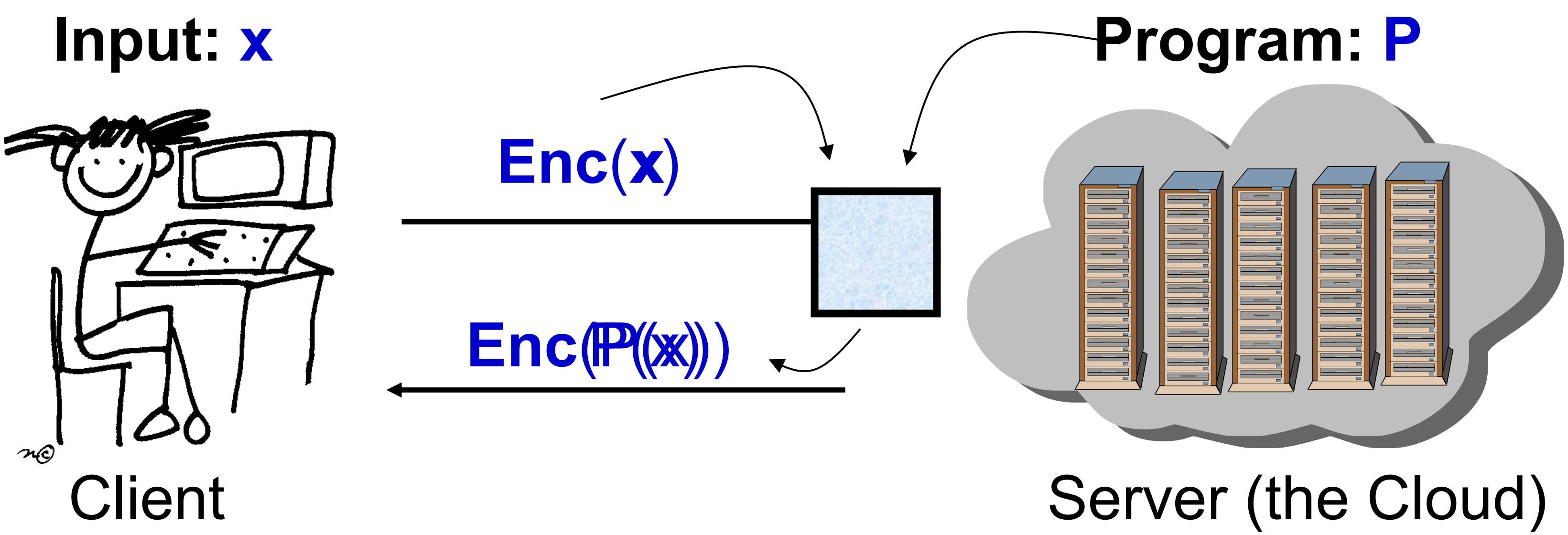
MIT 6.875 & Berkeley CS276

Foundations of Cryptography
Lecture 21

TODAY: Homomorphic Encryption

1. Secure Outsourcing



A Special Case: Encrypted Database Lookup

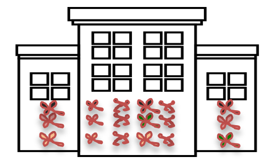
– also called "private information retrieval" (next lec)

2. Secure Collaboration (also called Secure Computation)

Genomics england

ID	Genome

Hospital



ID	Phenotype

“Parties learn the genotype-phenotype correlations and nothing else”

Homomorphic Encryption: Syntax

(can be either secret-key or public-key enc)

4-tuple of PPT algorithms $(Gen, Enc, Dec, Eval)$ s.t.

- $(sk, ek) \leftarrow Gen(1^n)$.

PPT Key generation algorithm generates a secret key as well as a (public) evaluation key.

- $c \leftarrow Enc(sk, m)$.

Encryption algorithm uses the secret key to encrypt message m.

- $c' \leftarrow Eval(ek, f, c)$.

Homomorphic evaluation algorithm uses the evaluation key to produce an “evaluated ciphertext” c'.

- $m \leftarrow Dec(sk, c)$.

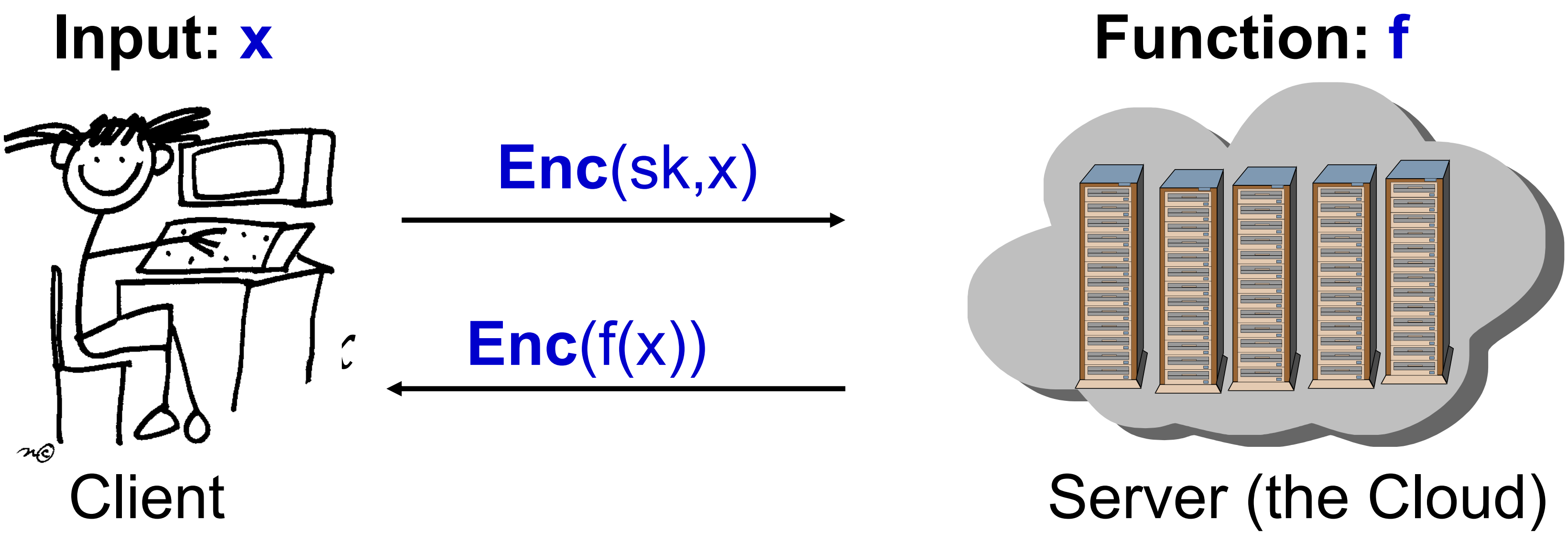
Decryption algorithm uses the secret key to decrypt ciphertext c.

Homomorphic Encryption: Correctness

$$Dec(sk, Eval(ek, f, Enc(x))) = f(x).$$

$$\begin{tikzcd}
x \arrow[rr, dashed] \arrow[d, "\boldsymbol{Enc}"'] & & f(x) \\
c \arrow[rr, "\boldsymbol{Eval(\cdot, f, \cdot)}"'] & & c' \arrow[u, "\boldsymbol{Dec}"']
\end{tikzcd}$$

Homomorphic Encryption: Security



Security against the curious cloud = standard **IND-security** of secret-key encryption

Key Point: Eval is an entirely public algorithm with public inputs.

Here is a homomorphic encryption scheme…

- $(sk, -) \leftarrow Gen(1^n)$.

Use any old secret key enc scheme.

- $c \leftarrow Enc(sk, m)$.

Just the secret key encryption algorithm…

- $c' \leftarrow Eval(ek, f, c)$.

Output c' = c || f. So Eval is basically the identity function!!

- $m \leftarrow Dec(sk, c')$.

Parse $c' = c||f$ as a ciphertext concatenated with a function description. Decrypt c and compute the function f.

This is correct and it is IND-secure.

Homomorphic Encryption: Compactness

The size (bit-length) of the evaluated ciphertext and the runtime of the decryption is *independent of* the complexity of the evaluated function.

A Relaxation: The size (bit-length) of the evaluated ciphertext and the runtime of the decryption *depends sublinearly on* the complexity of the evaluated function.

Big Picture: Two Steps to FHE

Leveled Secret-key Homomorphic Encryption:
Evaluate circuits of a-priori bounded depth *d*

"you give me a depth bound d, I will give you a homomorphic scheme that handles depth-d circuits…"

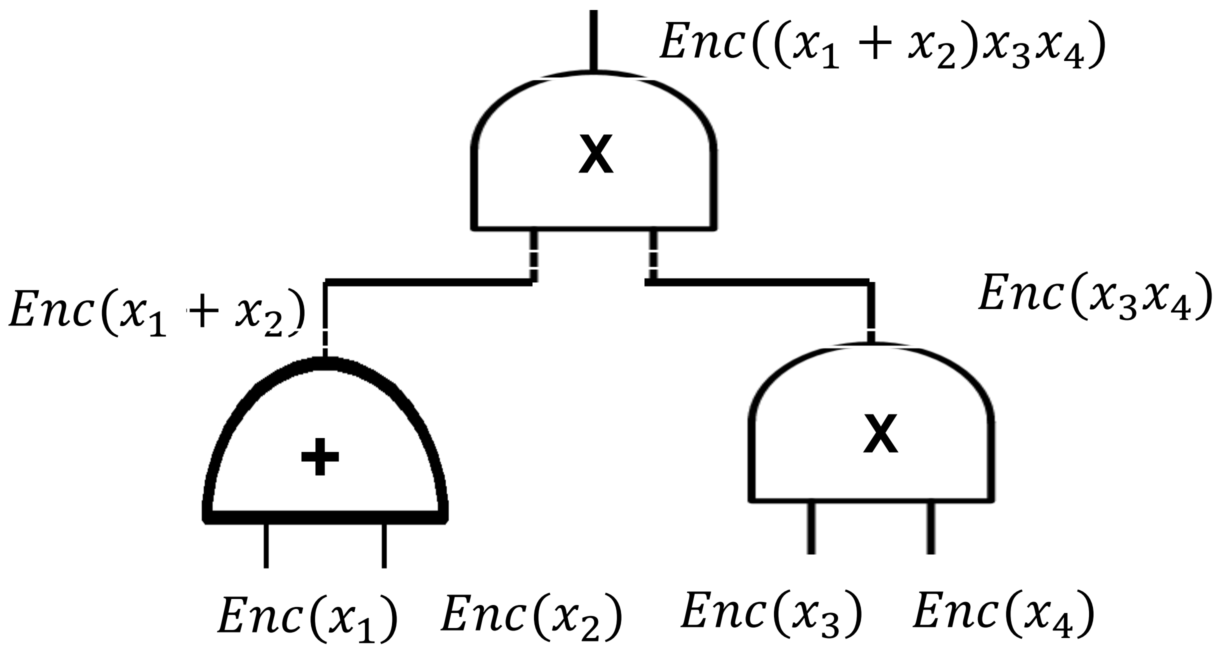
Bootstrapping Theorem:

From "circular secure" Leveled FHE to Pure FHE (at the cost of an additional assumption)

"I will give you homomorphic scheme that handles circuits of ANY size/depth"

How to Compute Arbitrary Functions

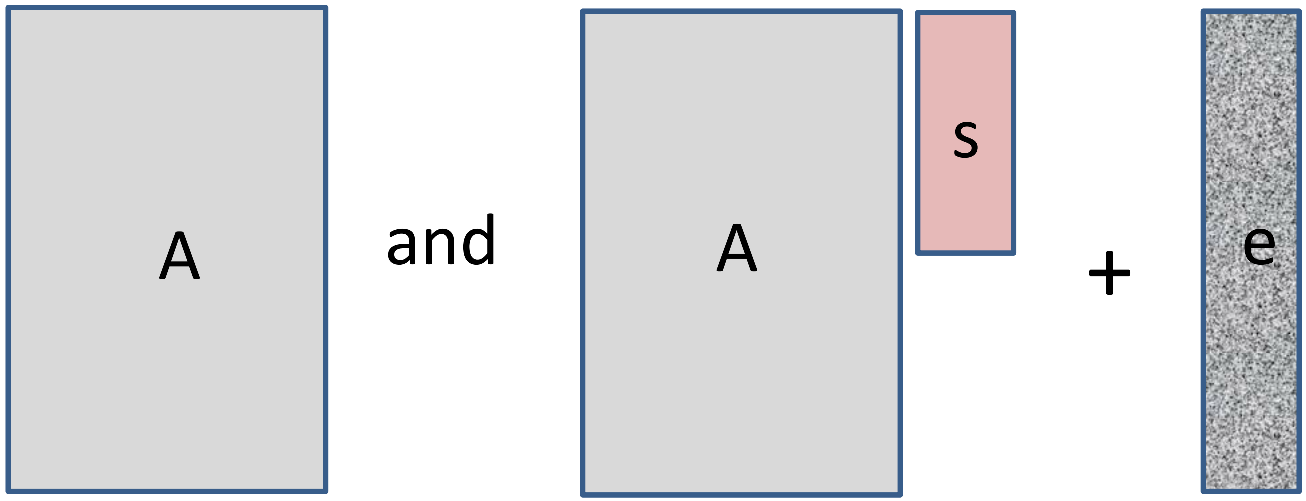
For us, programs = functions = Boolean circuits with XOR ($+ \ mod\ 2$) and AND ($\times \ mod\ 2$) gates.



Takeaway: If you can compute XOR and AND on encrypted bits, you can compute everything.

Learning with Errors (LWE)

Given:



GOAL: Find s.

<u>Parameters</u>: dimensions $\boldsymbol{n}$ and m, modulus $\boldsymbol{q}$, error distribution χ = uniform in some interval $[-\boldsymbol{B}, \ldots, \boldsymbol{B}]$.

A is chosen at random from $\mathbb{Z}_q^{m \times n}$, **s** from $\mathbb{Z}_q^n$ and **e** from χ^m.

Setting Parameters

Put together, we are safe with:

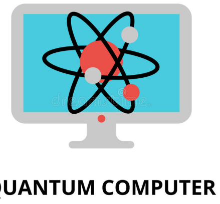
$n =$ security parameter ($\approx 1 - 10\text{K}$)

$m =$ arbitrary poly in n

$B =$ small poly in n, say $\sqrt{n}$

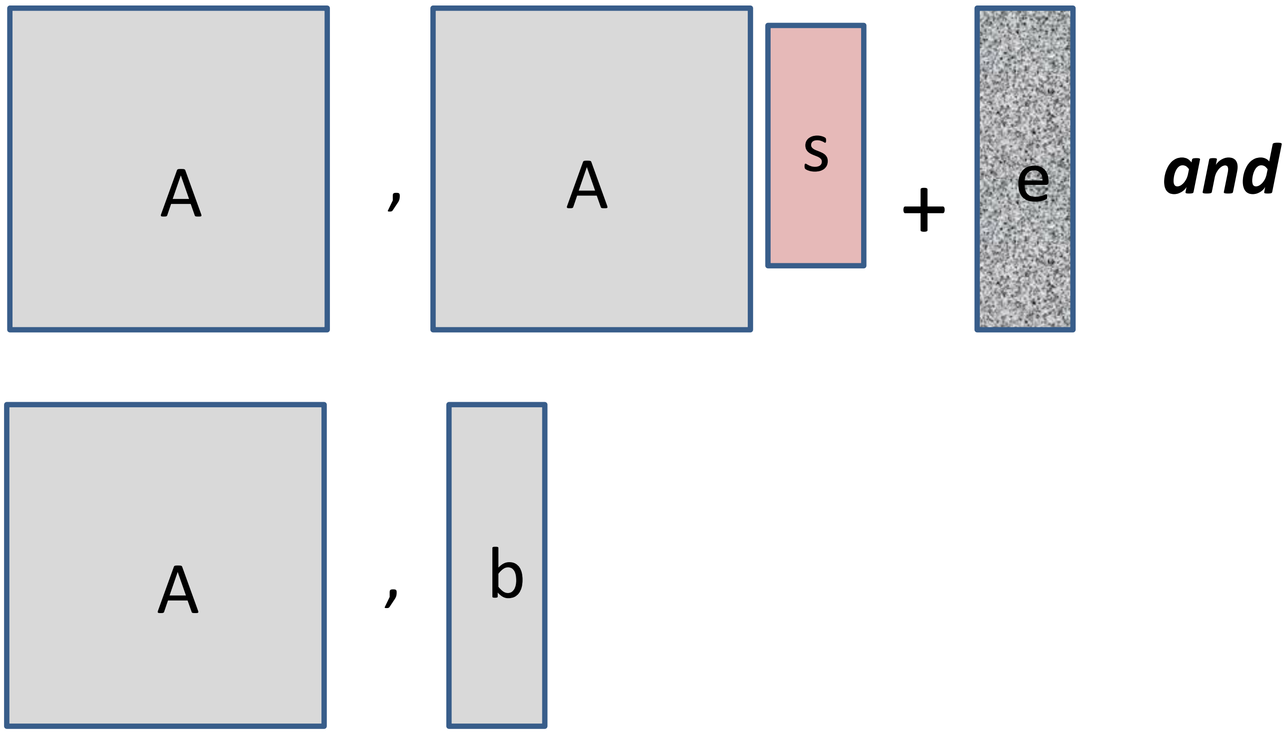
$q =$ poly in n, larger than B, and could be as large as sub-exponential, *say* $2^{n^{0.99}}$

even from quantum computers, AFAWK!



Decisional LWE

Can you distinguish between:



Theorem: "Decisional LWE is as hard as LWE".

Basic (Secret-key) Encryption

[Regev05]

n = security parameter, q = "small" modulus

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $\text{Enc}_{\mathbf{s}}(\mu)$: // $\mu \in \{0,1\}$
  - Sample uniformly random $\mathbf{a} \in Z_q^n$, "small" noise $e \in Z$
  - The ciphertext $\mathbf{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + \mu \lfloor q/2 \rfloor)$
- Decryption $\text{Dec}_{sk}(\mathbf{c})$: Output $\text{Round}_{q/2}(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q)$

  // correctness as long as $|e| < q/4$

New (Secret-key) Encryption: Take 1

- Private key: a vector **s** $\in Z_q^n$
- Private-key Encryption of a bit $m \in \{\mathbf{0}, \mathbf{1}\}$:

$$\mathbf{C} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{sA} \end{bmatrix} + m\, \boldsymbol{I}$$

($\boldsymbol{A}$ is random (n+1) X n matrix)

- Decryption:

$$\underbrace{[\mathbf{s} \,\|\, -1]}_{\text{Priv key = Eigenvector}} \quad \underbrace{\mathbf{C}}_{\text{Ciphertext matrix}} \quad = \quad \underbrace{m}_{\text{Message = Eigenvalue}} \; [\mathbf{s} \,\|\, -1] \quad (\text{mod } q)$$

🙁 **INSECURE!** Easy to solve linear equations.

New (Secret-key) Encryption: Take 1

$$\mathbf{t} \cdot \mathbf{C} = m \cdot \mathbf{t} \pmod{q}$$

$t = [s \,||\, -1]$

► Homomorphic addition: $\mathbf{C_1} + \mathbf{C_2}$

– t is an eigenvector of C_1+C_2 with eigenvalue m_1+m_2

► Homomorphic multiplication: $\mathbf{C_1 C_2}$

– t is an eigenvector of C_1C_2 with eigenvalue m_1m_2

Proof: $\mathbf{t} \cdot \mathbf{C_1}\, \mathbf{C_2} = (m_1 \cdot \mathbf{t}) \cdot \mathbf{C_2} = m_1 \cdot m_2 \cdot \mathbf{t}$

But, remember, the scheme is insecure?

Key idea: fix insecurity while retaining homomorphism.

New (Secret-key) Encryption: Take 2

- Private key: a vector **s** $\in Z_q^n$
- Private-key Encryption of a bit $m \in \{0,1\}$:

$$\mathbf{C} = \begin{bmatrix} A \\ sA + e \end{bmatrix} + m\,I$$

(A is random (n+1) X n matrix)

- Decryption:

$$\underbrace{[\mathbf{s} \,||\, \mathbf{-1}]}_{\text{Priv key = Approx Eigenvector}} \;\; \underbrace{\mathbf{C}}_{\text{Ciphertext matrix}} \;\approx\; \underbrace{m}_{\text{Message = Approx Eigenvalue}} \,[\mathbf{s} \,||\, \mathbf{-1}] \pmod q$$

🙂 **CPA-secure by LWE.**

New (Secret-key) Encryption: Take 2

t . C = m . t + e (mod q)

t = [s || -1]

► Homomorphic addition: $\mathbf{C_1} + \mathbf{C_2}$

$$
\begin{aligned}
\vec{t} \cdot (C_1 + C_2) &= \vec{t}C_1 + \vec{t}C_2 \\
&= m_1\vec{t} + \vec{e}_1 + m_2\vec{t} + \vec{e}_2 \\
&= (m_1 + m_2)\vec{t} + (\vec{e}_1 + \vec{e}_2) \\
&\approx (m_1 + m_2)\vec{t}
\end{aligned}
$$

Noise grows a little

New (Secret-key) Encryption: Take 2

t . C = m . t + e (mod q)

t = [s || -1]

► Homomorphic multiplication: $\mathbf{C_1 C_2}$

Can also use $C_2 C_1$

$$\vec{t} \cdot (C_1 \cdot C_2) = (m_1 \vec{t} + \vec{e}_1) C_2$$
$$= m_1 \vec{t} C_2 + \vec{e}_1 C_2$$
$$= m_1 (m_2 \vec{t} + \vec{e}_2) + \vec{e}_1 C_2$$
$$= m_1 m_2 \vec{t} + \underbrace{m_1 \vec{e}_2 + \vec{e}_1 C_2}_{\vec{e}_{mult}}$$

Noise grows. **Need C_2 to be small! How?!**

Aside: Binary Decomposition

Break each entry in C into its binary representation

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

Small entries like we wanted!

Consider the "reverse" operation:

$$k \ \underbrace{\overbrace{\left[\qquad\qquad\qquad\right]}^{k \log q}}_{G} \cdot bits(C) = C \quad \Longrightarrow \quad \boxed{\vec{t} \cdot C = \vec{t} \cdot G \cdot G^{-1}(C)}$$

Denote: $G^{-1}(C)$ which has "small" entries

New (Secret-key) Encryption: Take 3

- Private key: a vector **s** $\in \boldsymbol{Z_q^n}$
- Private-key Encryption of a bit $m \in \{\mathbf{0}, \mathbf{1}\}$:

$$\mathbf{C} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{sA} + \boldsymbol{e} \end{bmatrix} + m\, G$$

($\boldsymbol{A}$ is random (n+1) X n log q matrix)

- Decryption:

$$\underbrace{[\mathbf{s} \,||\, \mathbf{-1}]}_{\text{Priv key = Approx Eigenvector}} \quad \underbrace{\mathbf{C}}_{\text{Ciphertext matrix}} \quad \approx \quad \underbrace{m}_{\text{Message = Approx “Eigenvalue”}} \,[\mathbf{s} \,||\, \mathbf{-1}]\, \mathbf{G} \quad (\text{mod } q)$$

🙂 **Still CPA-secure by LWE.**

New (Secret-key) Encryption: Take 3

t . C = m . t . G + e (mod q)

t = [s || -1]

▶ Homomorphic multiplication: $C_{mult} = C_1 \cdot G^{-1}(C_2)$

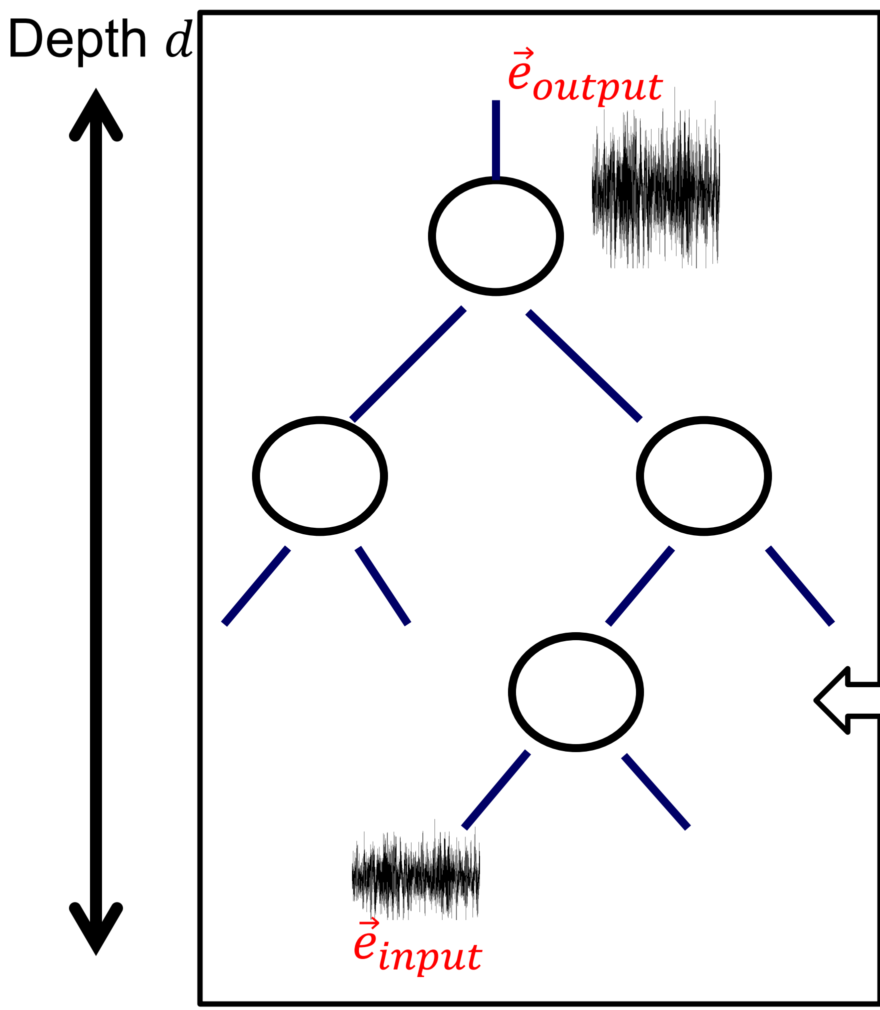
$$\begin{aligned}\vec{s} \cdot C_1 \cdot G^{-1}(C_2) &= (\vec{e}_1 + m_1 \cdot \vec{s} \cdot G) \cdot G^{-1}(C_2) \\ &= \vec{e}_1 \cdot G^{-1}(C_2) + m_1 \cdot \vec{s} \cdot G \cdot G^{-1}(C_2) \\ &= \vec{e}_1 \cdot G^{-1}(C_2) + m_1 \cdot \vec{s} \cdot C_2 \\ &= \vec{e}_1 \cdot G^{-1}(C_2) + m_1 \cdot (\vec{e}_2 + m_2 \cdot \vec{s} \cdot G) \\ &= \underbrace{(\vec{e}_1 \cdot G^{-1}(C_2) + m_1 \cdot \vec{e}_2)}_{\vec{e}_{mult}} + m_1 m_2 \cdot \vec{s} \cdot G\end{aligned}$$

$$\|\vec{e}_{mult}\| \leq n \log q \cdot \|\vec{e}_1\| + m_1 \cdot \|\vec{e}_2\| \leq (n \log q + 1) \cdot \max\{\|\vec{e}_1\|, \|\vec{e}_2\|\}$$

$Let\ N = n \log q$

Homomorphic Circuit Evaluation

Noise grows during homomorphic eval



$\|\vec{e}_{output}\| \leq (N+1)^d \cdot B_0 \approx N^d B_0$

$\Rightarrow$ Decryptable if $q \gg N^d B_0$.

(for security: $q \ll 2^n$)

So this can support $d \approx n^{0.99}$

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

$\|\vec{e}_{input}\| \leq B_0$

Big Picture: Two Steps to FHE

Leveled Secret-key Homomorphic Encryption:
Evaluate circuits of a-priori bounded depth *d*

"you give me a depth bound d, I will give you a homomorphic scheme that handles depth-d circuits…"

Bootstrapping Theorem:

From "circular secure" Leveled FHE to Pure FHE (at the cost of an additional assumption)

"I will give you homomorphic scheme that handles circuits of ANY size/depth"

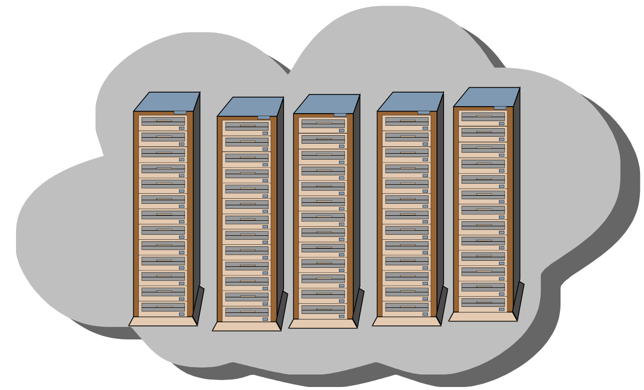
From Leveled to Fully Homomorphic

Input: x

Client

Enc(sk,x) →

Function: f



Server (the Cloud)

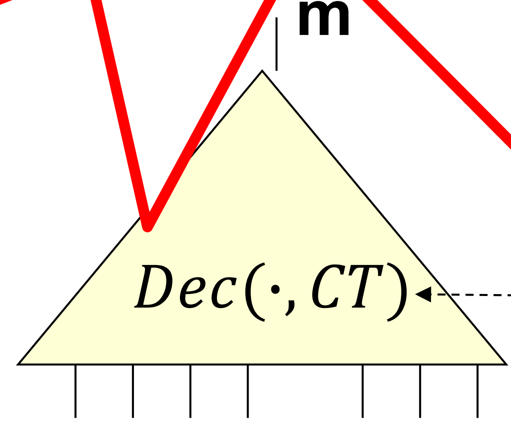
The cloud keeps homomorphically computing, but after a certain depth, the ciphertext is too noisy to be useful. What to do?

Idea: "Bootstrapping"!

Bo… y

“Best Pos… n!

But the evaluator/cloud does not have SK!



“Noisele… ciphertext”

“Very Noisy” ciphertext

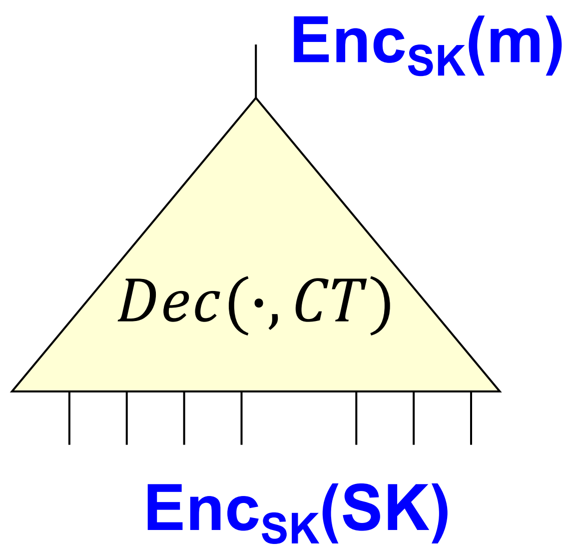
Decryption Circuit

Bootstrapping, Concretely

Next Best = Homomorphic Decryption!

*  Assume server knows **$\mathbf{ek} = \mathbf{Enc_{SK}(SK)}$**.

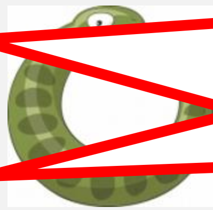
(OK assuming the scheme is "circular secure")



Bootstrapping, Concretely

Next Best = Homomorph... Decryption!

*

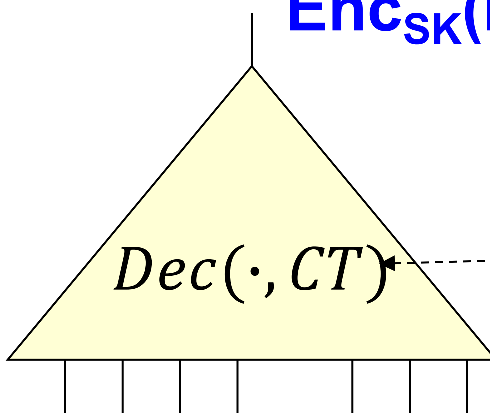


...k(SK).

(C... ...ircular secure")

B_{dec} Independent of B_{input}

$Enc_{SK}(m)$ — **Noise = B_{dec}**



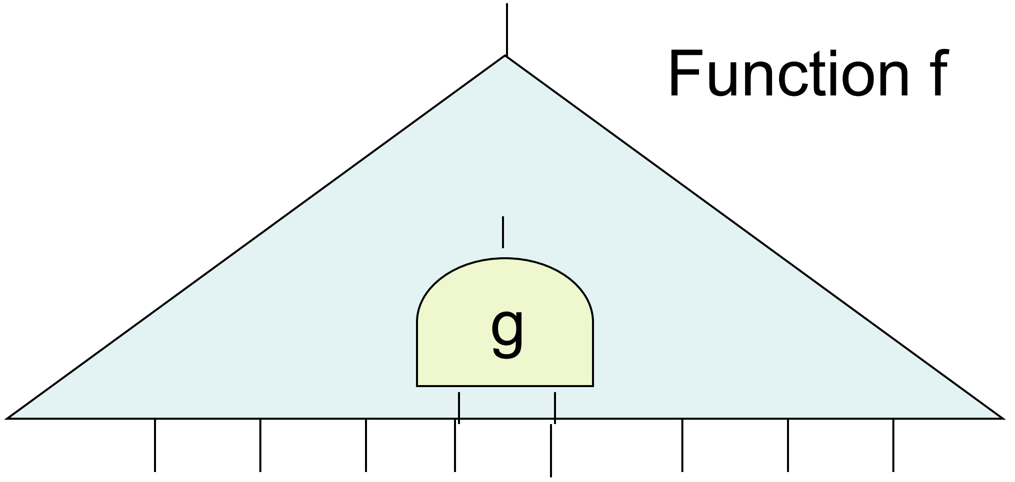
Noise = B_{input}

$Enc_{SK}(SK)$

Wrap Up: Bootstrapping

Assume Circular Security:

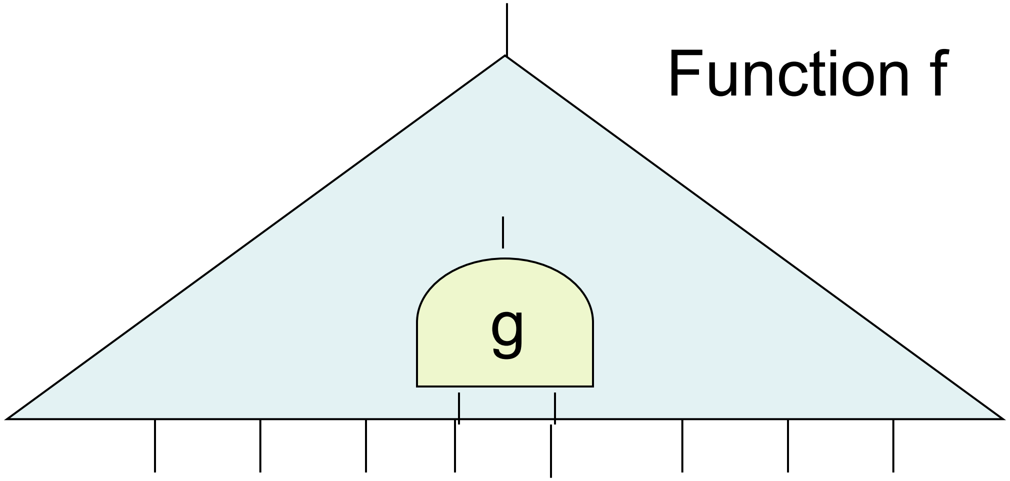
Evaluation key is $\mathbf{Enc_{SK}(SK)}$



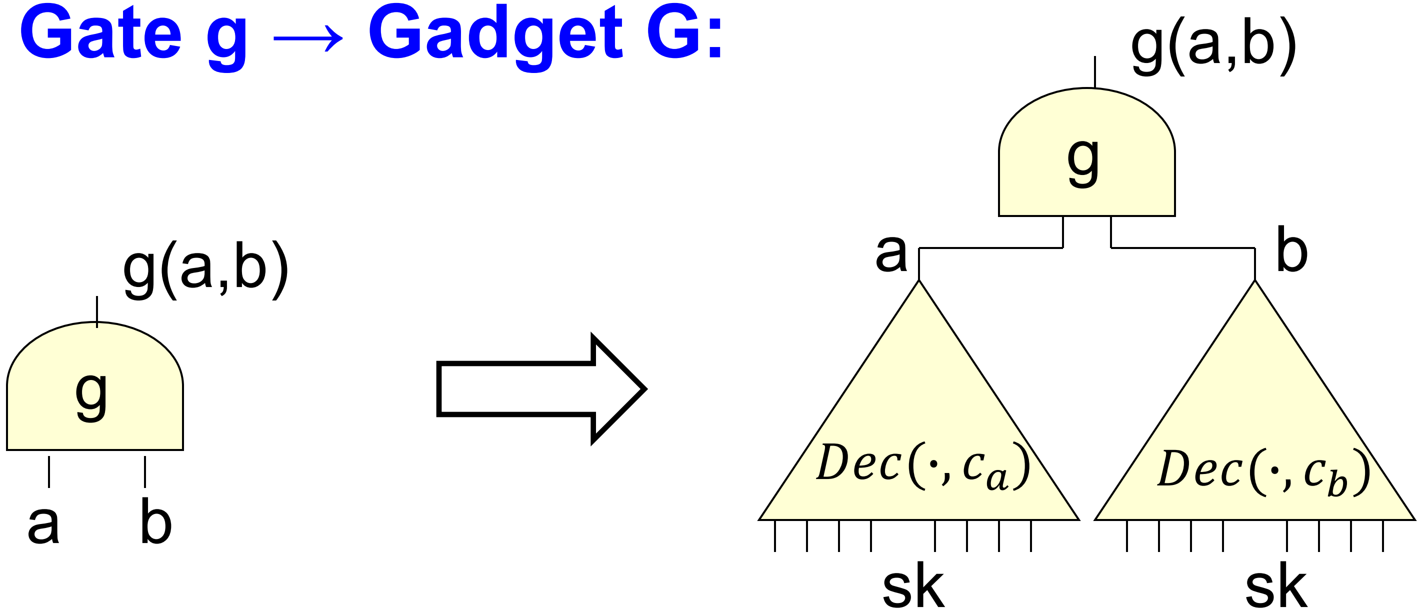
Wrap Up: Bootstrapping

Assume Circular Security:

Evaluation key is **$\mathrm{Enc}_{SK}(SK)$**



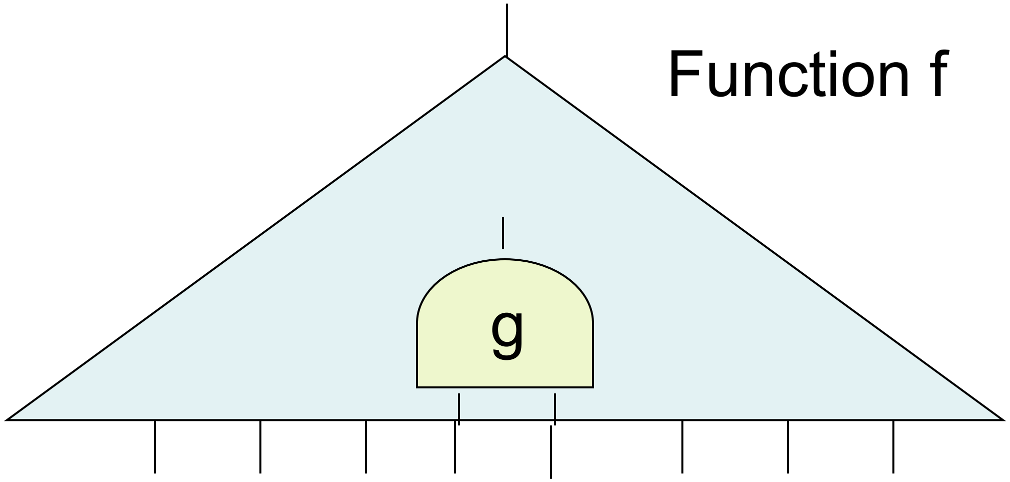
Each Gate g → Gadget G:



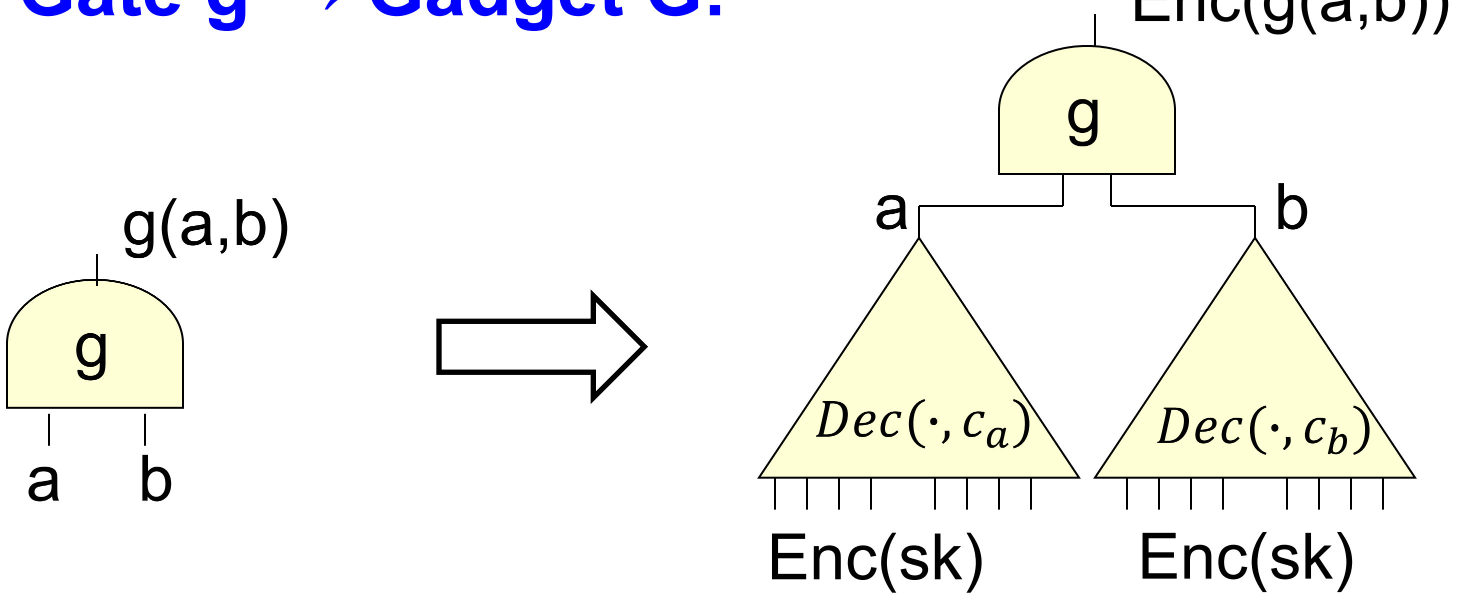
Wrap Up: Bootstrapping

Assume Circular Security:

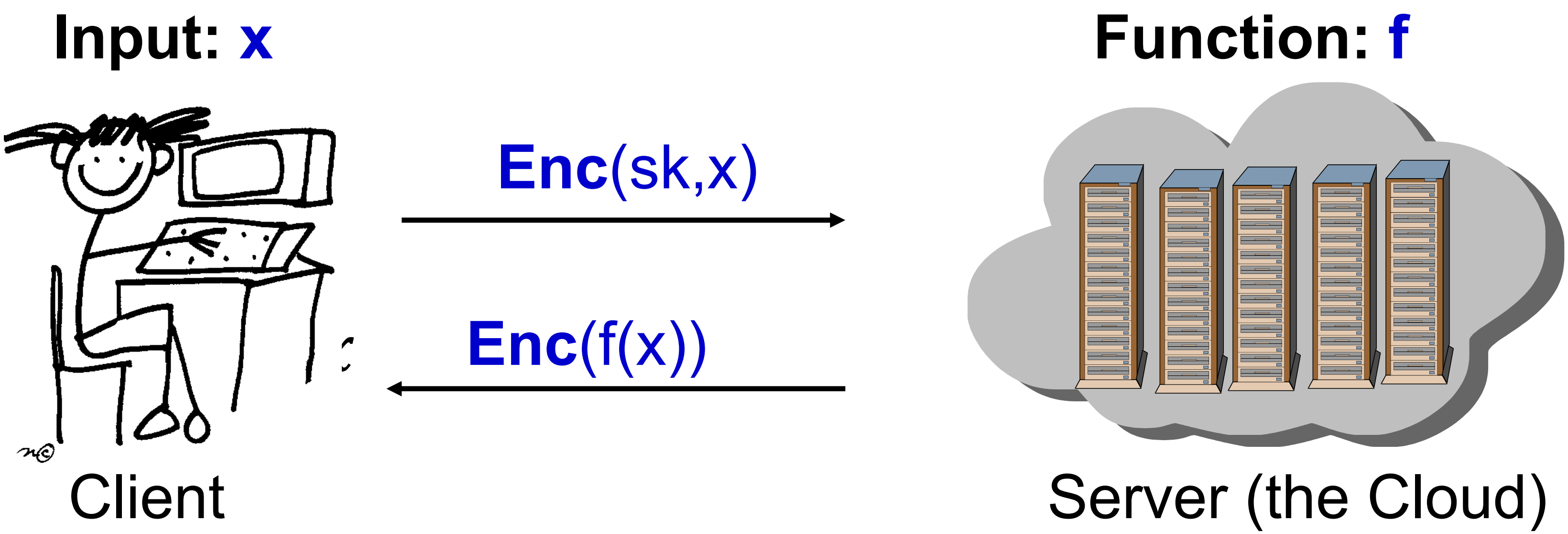
Evaluation key is **$\mathrm{Enc}_{SK}(SK)$**



Each Gate g → Gadget G:



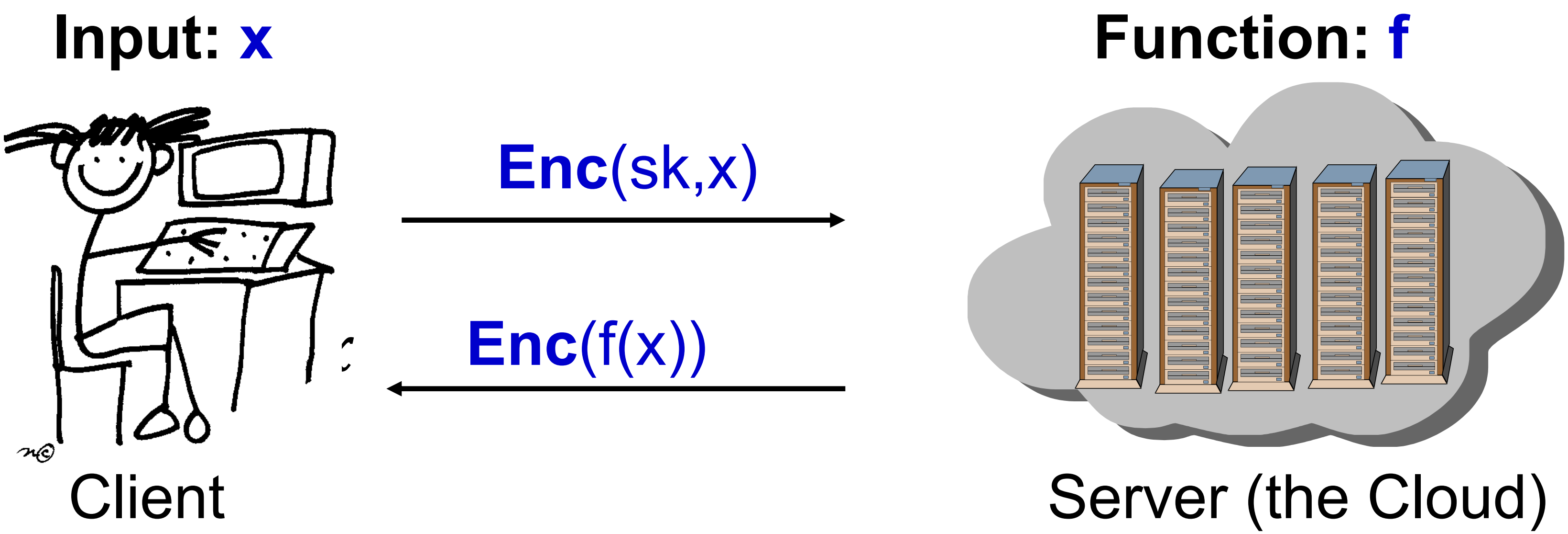
How about Function Privacy?



Security against the curious cloud = standard **IND-security** of secret-key encryption

Security against a curious user?

Function Privacy



Function Privacy: Enc(f(x)) reveals no more information (about f) than f(x).

HOMOMORPHIC ENCRYPTION IN PRACTICE

DARPA $60M investment [2012-17].

Many Open Source Libraries.

PALISADE	SEAL
HELib	HEEAN

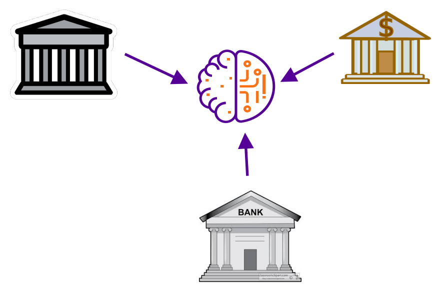
APPLICATIONS of HOMOMORPHIC ENCRYPTION

Healthcare
Applying genomic analysis to 1K patients
13 seconds

Winner of the 2018 iDash International Homomorphic Encryption competition

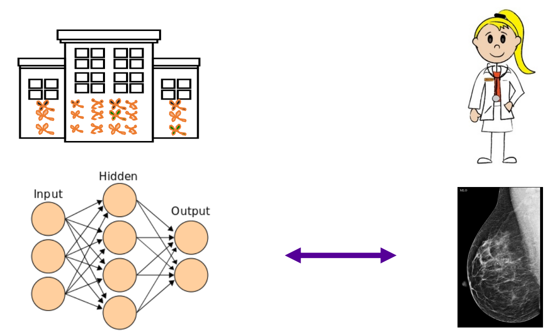
Collaboration with Dana Farber and Duality Technologies.

Financial
Benchmarking cyberrisk on 1M records
12 seconds



Collaboration with Andrew Lo@Sloan and Danny Weitzner@CSAIL Internet Policy Research Initiative.

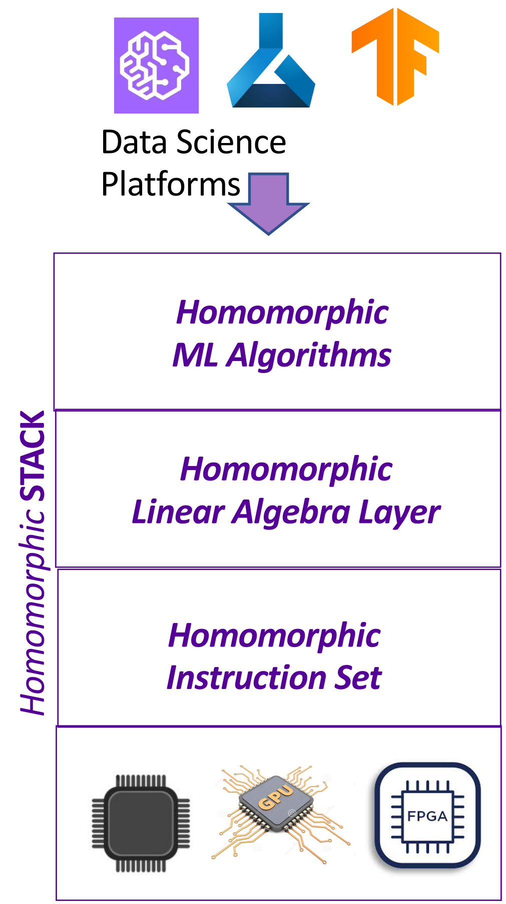
Medical Imaging
Breast density detection on encrypted mammograms
60 seconds



Collaboration with Regina Barzilay@CSAIL and Anantha Chandrakasan@EECS.

Synergy of Algorithms & Data Science & HPC & Crypto

THE DREAM



Many Secure Computing Startups.

Standardization Efforts.

homomorphicencryption.org

Next Lecture:
Homomorphic Encryption and Database Lookup