Somewhat Homomorphic Encryption from Linear Homomorphism and Sparse LPN

Henry Corrigan-Gibbs

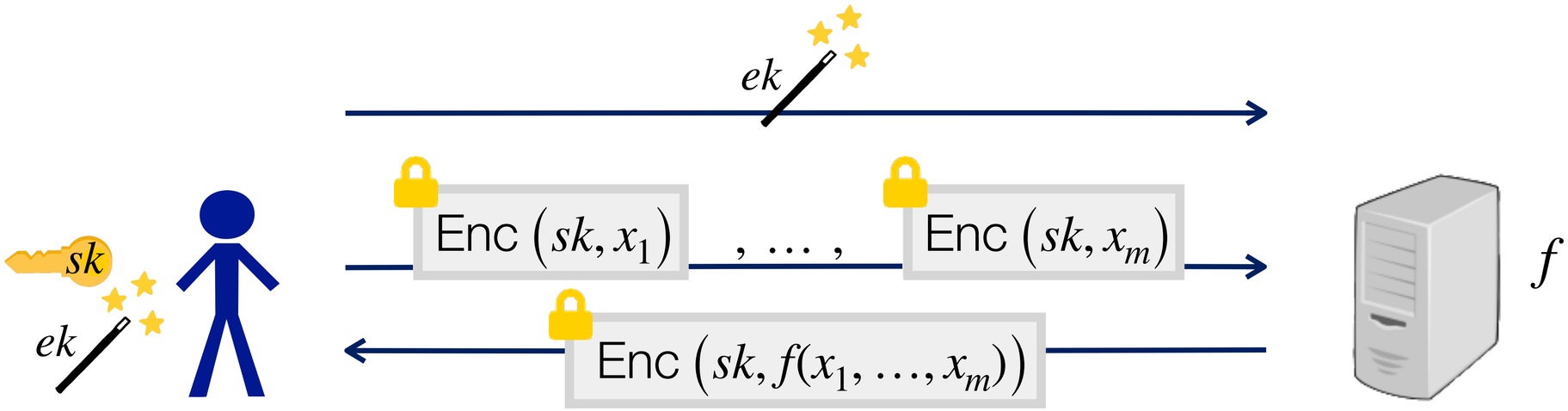
Alexandra Henzinger

Yael Kalai

Vinod Vaikuntanathan

Homomorphic Encryption for a class of computations C: for all $f \in C$,

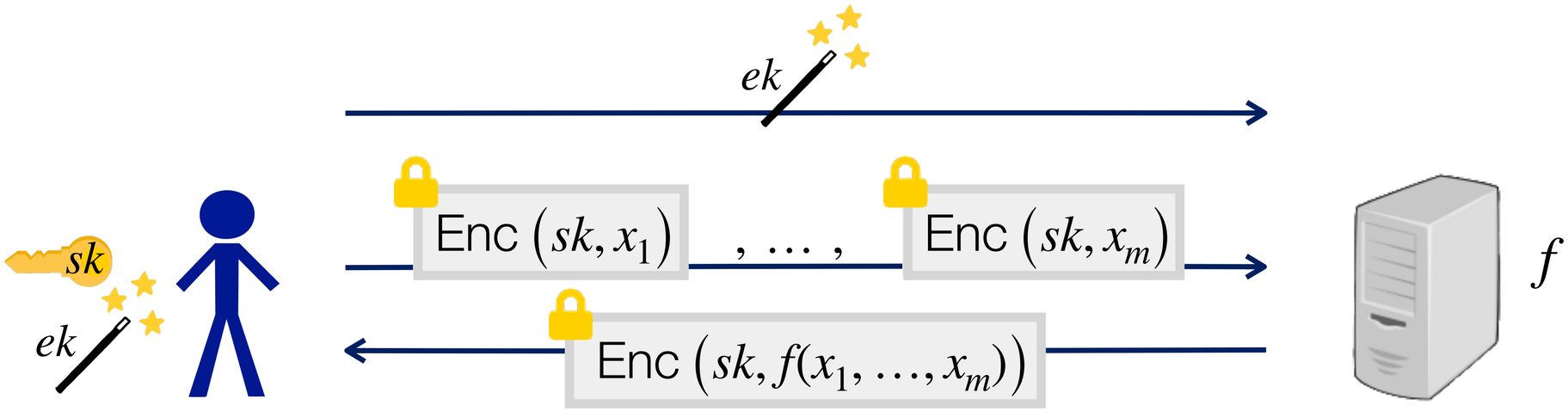
[RAD78]



1. **Correct.** With , Enc $\left(sk, f(x_1, \ldots, x_m)\right)$ decrypts to $f(x_1, \ldots, x_m)$ with good probability.
2. **Semantically secure.** The ciphertexts and ek reveal nothing about $x_1, \ldots, x_m$.
3. **Compact.** The bitlength of each ciphertext does not grow with $|f|$.

Homomorphic Encryption for a class of computations C: for all $f \in C$,

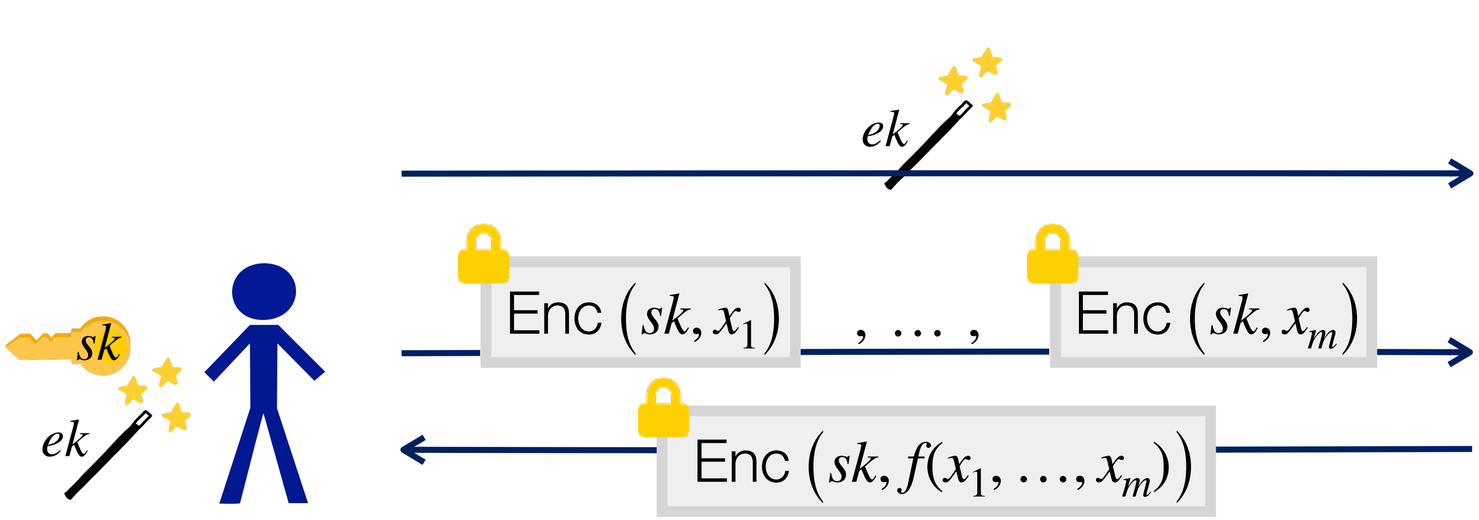
[RAD78]

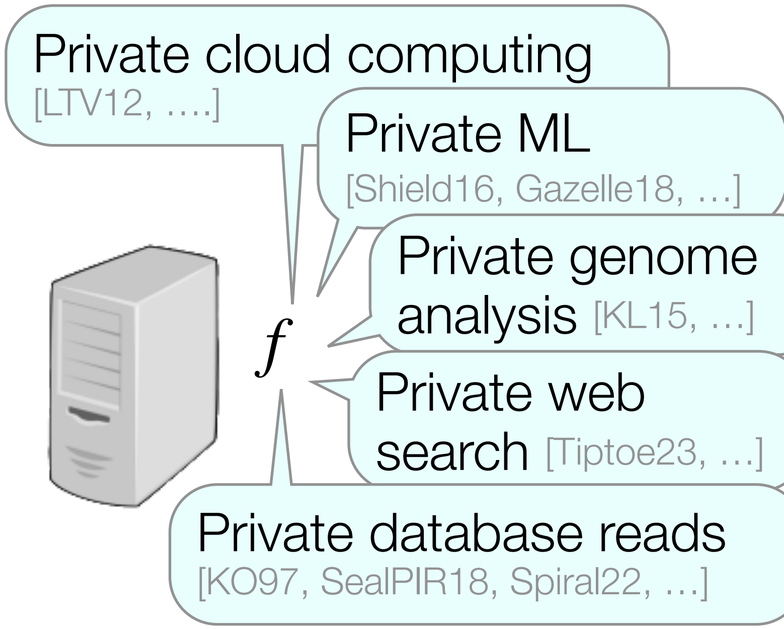


1. **Correct.** With sk, Enc $\left(sk, f(x_1, \ldots, x_m)\right)$ decrypts to $f(x_1, \ldots, x_m)$ with good probability.
2. **Semantically secure.** The ciphertexts and ek reveal nothing about $x_1, \ldots, x_m$.
3. **Compact.** The bitlength of each ciphertext does not grow with $|f|$.

Homomorphic Encryption for a class of computations C: for all $f \in C$,

[RAD78]





1. **Correct.** With sk, Enc $\left(sk, f(x_1, \ldots, x_m)\right)$ decrypts to $f(x_1, \ldots, x_m)$ with good probability.
2. **Semantically secure.** The ciphertexts and ek reveal nothing about $x_1, \ldots, x_m$.
3. **Compact.** The bitlength of each ciphertext does not grow with $|f|$.

Fully Homomorphic Enc

computes any number of adds and muls

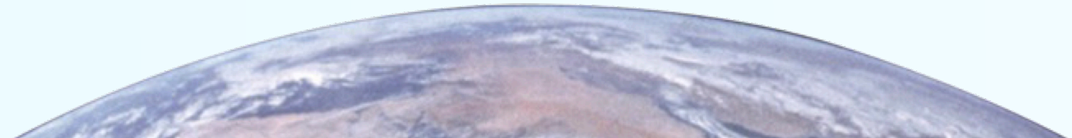
Somewhat Homomorphic Enc

Linearly Homomorphic Enc

computes any number of adds

Private-Key Enc

can’t compute on ciphertexts



Fully Homomorphic Enc

computes any number of adds and muls

Somewhat Homomorphic Enc

Linearly Homomorphic Enc

computes any number of adds

Private-Key Enc

can’t compute on ciphertexts

One-way functions

Fully Homomorphic Enc

computes any number of adds and muls

Somewhat Homomorphic Enc

Linearly Homomorphic Enc

computes any number of adds

Number-theoretic crypto

Private-Key Enc

can't compute on ciphertexts

One-way functions

Fully Homomorphic Enc

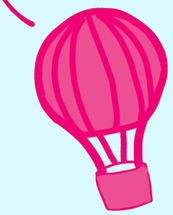
computes any number of adds and muls

Somewhat Homomorphic Enc

Computes small-length branching programs [IP07,…]

Linearly Homomorphic Enc

computes any number of adds



Number-theoretic crypto

Private-Key Enc

can't compute on ciphertexts

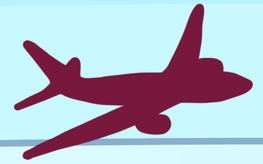
One-way functions

Fully Homomorphic Enc

computes any number of adds and muls

Somewhat Homomorphic Enc

Pairing crypto: computes one mul and many adds [BGN05]



Computes small-length branching programs [IP07,…]

Linearly Homomorphic Enc

computes any number of adds

Number-theoretic crypto

Private-Key Enc

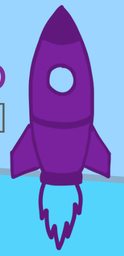
can't compute on ciphertexts

One-way functions

Fully Homomorphic Enc

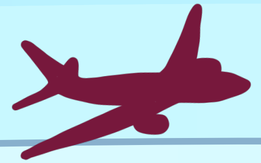
computes any number of adds and muls

Lattice crypto
[G09,…]



Somewhat Homomorphic Enc

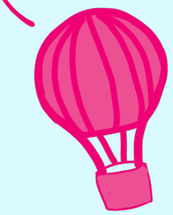
Pairing crypto: computes one mul and many adds [BGN05]



Computes small-length branching programs [IP07,…]

Linearly Homomorphic Enc

computes any number of adds



Number-theoretic crypto

Private-Key Enc

can't compute on ciphertexts

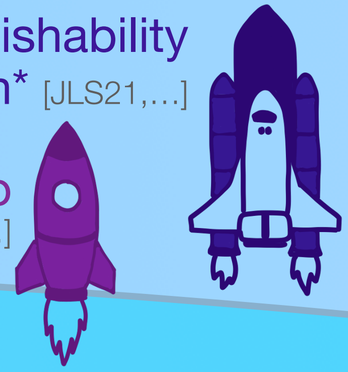
One-way functions

Fully Homomorphic Enc

computes any number of adds and muls

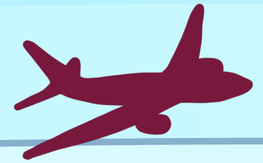
Indistinguishability obfuscation* [JLS21,…]

Lattice crypto [G09,…]



Somewhat Homomorphic Enc

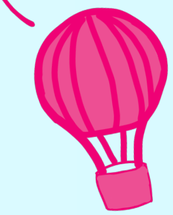
Pairing crypto: computes one mul and many adds [BGN05]



Computes small-length branching programs [IP07,…]

Linearly Homomorphic Enc

computes any number of adds



Number-theoretic crypto

Private-Key Enc

can't compute on ciphertexts

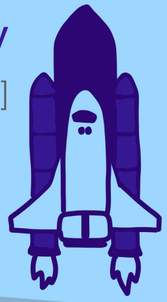
One-way functions

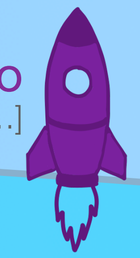
Fully Homomorphic Enc

computes any number of adds and muls

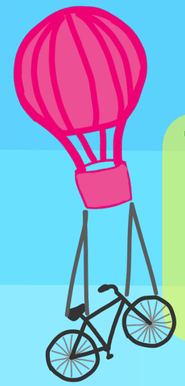
Indistinguishability obfuscation* [JLS21,…]

Lattice crypto [G09,…]



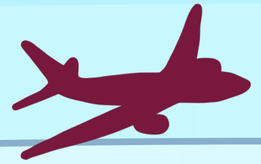


Somewhat Homomorphic Enc



This work: computes a few muls and many adds, using
- linearly homomorphic enc and
- private-key crypto

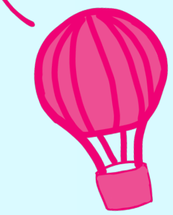
Pairing crypto: computes one mul and many adds [BGN05]



Computes small-length branching programs [IP07,…]

Linearly Homomorphic Enc

computes any number of adds



Number-theoretic crypto

Private-Key Enc

can't compute on ciphertexts

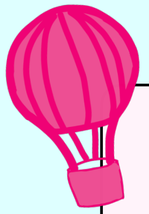
One-way functions

This talk

Sparse LPN

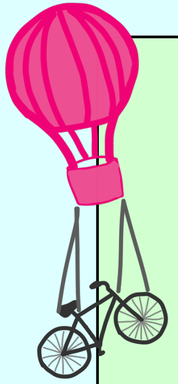
with modulus $q \geq 3$

+



Linearly Homomorphic Encryption

over $\mathbb{F}_q$ (known from DDH/DCR)



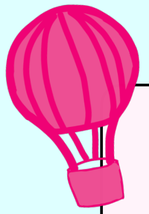
Somewhat Homomorphic Encryption that, for any $n = \text{poly}(\lambda)$ chosen during encryption, can perform $o(\log n)$ muls followed by n adds over $\mathbb{F}_q$

This talk

Sparse LPN

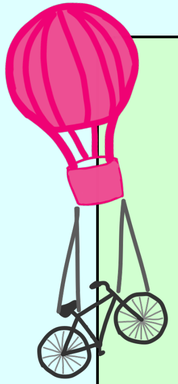
with modulus $q \geq 3$

+



Linearly Homomorphic Encryption

over $\mathbb{F}_q$ (known from DDH/DCR)



Somewhat Homomorphic Encryption that, for any $n = \text{poly}(\lambda)$ chosen during encryption, can perform $o(\log n)$ muls followed by n adds over $\mathbb{F}_q$

Background: Sparse Learning Parities with Noise

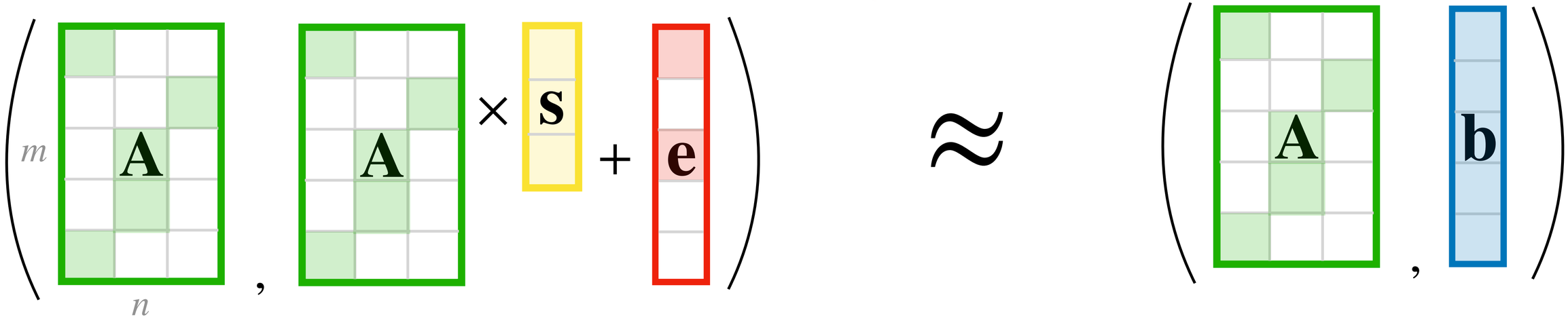
[Ale03, AIK06, IKOS08, ABW10, …]

$\mathbf{A}$: random in $\mathbb{F}_q^{m \times n}$ with each row k-sparse

$\mathbf{s} \xleftarrow{R} \mathbb{F}_q^n$

$\mathbf{e}$: each entry $\begin{cases} \text{random, with prob. } n^{-0.1} \\ 0, \text{ else} \end{cases}$

$\mathbf{b} \xleftarrow{R} \mathbb{F}_q^m$



Formally: $n, m = \text{poly}(\lambda)$, $q \leq \exp(\lambda)$ is a prime, the sparsity k is $\omega(1)$

Background: Sparse Learning Parities with Noise

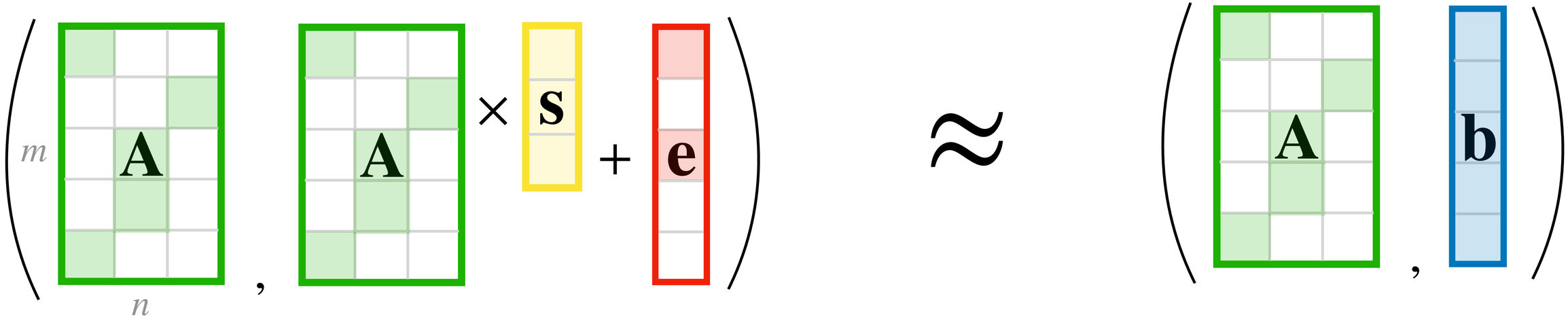
[Ale03, AIK06, IKOS08, ABW10, …]

$\mathbf{A}$: random in $\mathbb{F}_q^{m \times n}$ with each row k-sparse

$\mathbf{s} \xleftarrow{R} \mathbb{F}_q^n$

$\mathbf{e}$: each entry $\begin{cases} \text{random, with prob. } n^{-0.1} \\ 0, \text{ else} \end{cases}$

$\mathbf{b} \xleftarrow{R} \mathbb{F}_q^m$



In higher-noise settings: LPN $\Longrightarrow$ sparse LPN [JLS24, BBTV24]

In this setting: sparse LPN $\Longrightarrow$ constant-overhead PRGs, homomorphic secret sharing [IKOS08, DIJL23]

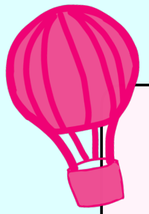
Formally: $n, m = \text{poly}(\lambda)$, $q \leq \exp(\lambda)$ is a prime, the sparsity k is $\omega(1)$

This talk

Sparse LPN

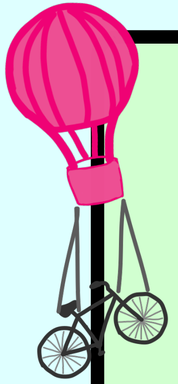
with modulus $q \geq 3$

+



Linearly Homomorphic Encryption

over $\mathbb{F}_q$ (known from DDH/DCR)



Somewhat Homomorphic Encryption that, for any $n = \text{poly}(\lambda)$ chosen during encryption, can perform $o(\log n)$ muls followed by n adds over $\mathbb{F}_q$

Design Ideas

Step 1: Use Sparse LPN to homomorphically add and multiply [GSW13, DIJL23]

- \- No compactness: ciphertexts are huge
- \+ Decryption is simple: it is a linear function in the secret key

Step 2: Use Linearly Homomorphic Encryption to shrink the ciphertexts [Gen09]

Design Ideas

Step 1: Use Sparse LPN to homomorphically add and multiply [GSW13, DIJL23]

- \- No compactness: ciphertexts are huge
- \+ Decryption is simple: it is a linear function in the secret key

Step 2: Use Linearly Homomorphic Encryption to shrink the ciphertexts [Gen09]

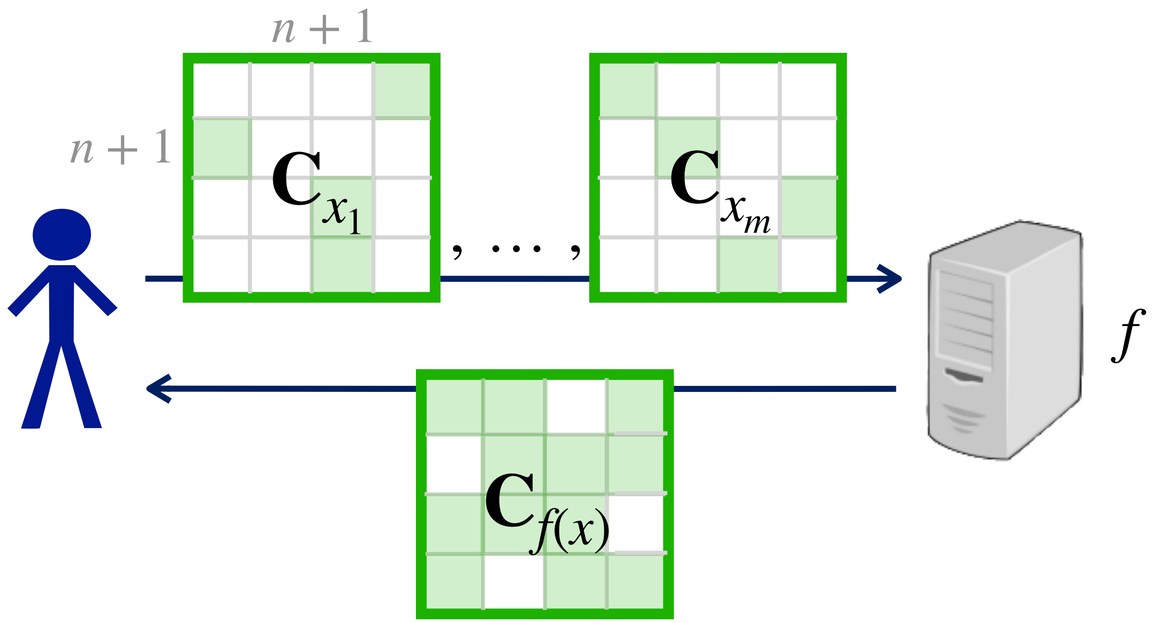
Design Ideas

Step 1: Use Sparse LPN to homomorphically add and multiply [GSW13, DIJL23]

- \- No compactness: ciphertexts are huge
- \+ Decryption is simple: it is a linear function in the secret key

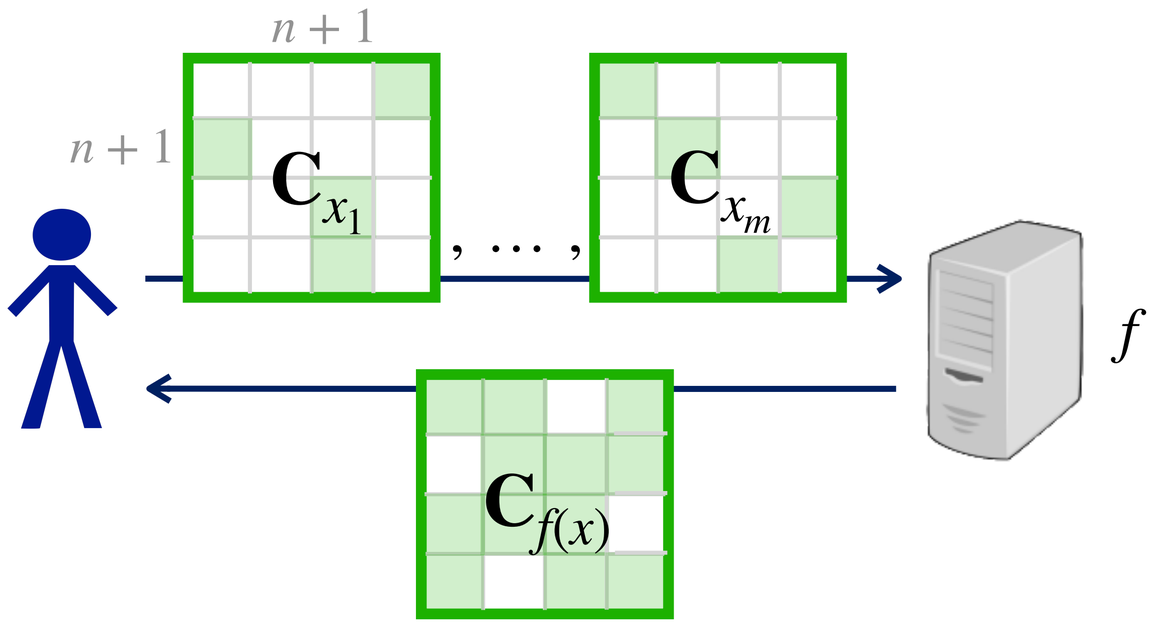
Step 2: Use Linearly Homomorphic Encryption to shrink the ciphertexts [Gen09]

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

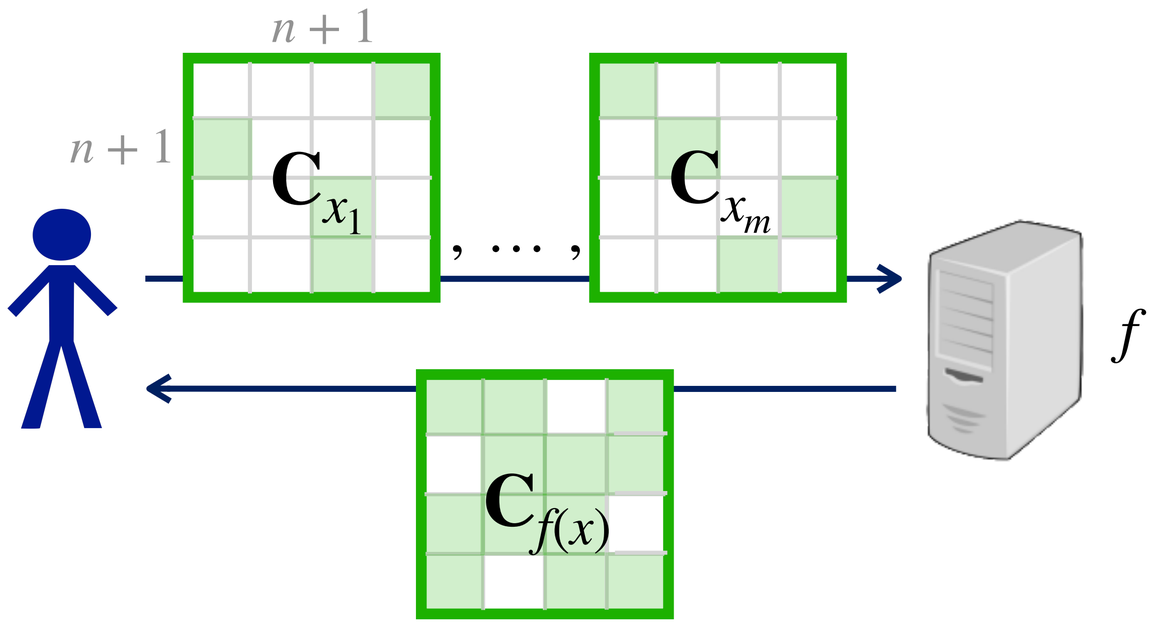
The key equation [GSW13]

Take sk to be a random vector $\mathbf{s} \in \mathbb{F}_q^n$.

Encrypt $x \in \mathbb{F}_q$ into a sparse matrix $\mathbf{C}_x \in \mathbb{F}_q^{(n+1)\times(n+1)}$, of which x is a "noisy" eigenvalue with sparse errors $\mathbf{e}$:

$$\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$$

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

The key equation [GSW13]

Take sk to be a random vector $\mathbf{s} \in \mathbb{F}_q^n$.

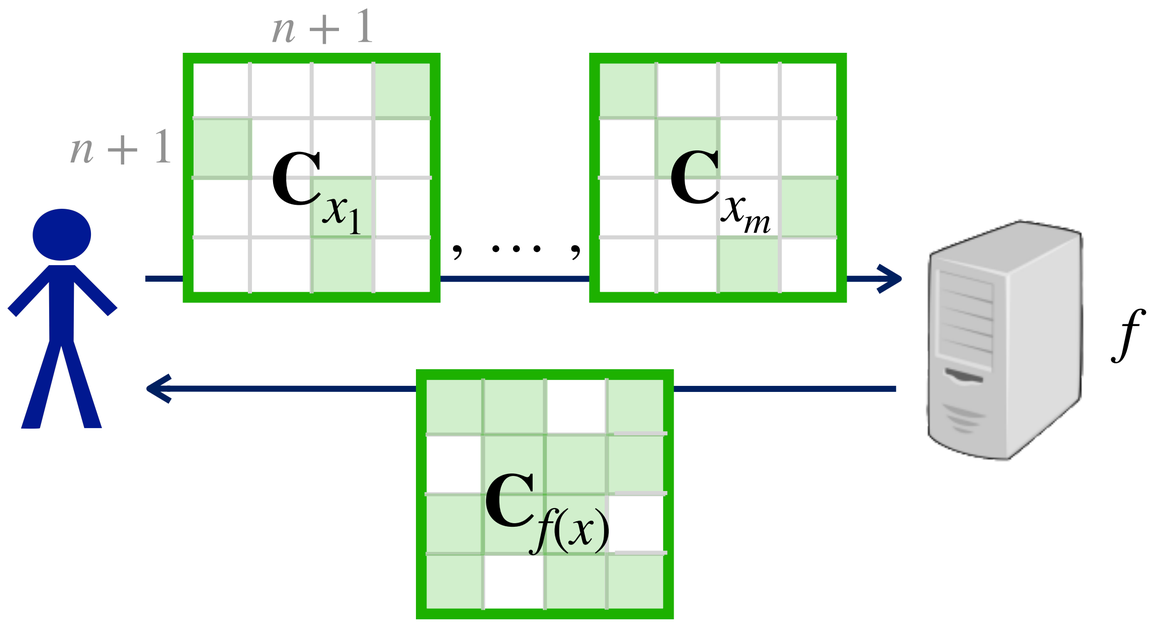
Encrypt $x \in \mathbb{F}_q$ into a sparse matrix $\mathbf{C}_x \in \mathbb{F}_q^{(n+1)\times(n+1)}$, of which x is a "noisy" eigenvalue with sparse errors $\mathbf{e}$:

$$\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$$

How?

$$\mathbf{C}_x = \left(\mathbf{A}, \mathbf{A} \times \mathbf{s} + \mathbf{e} + x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \right)$$

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

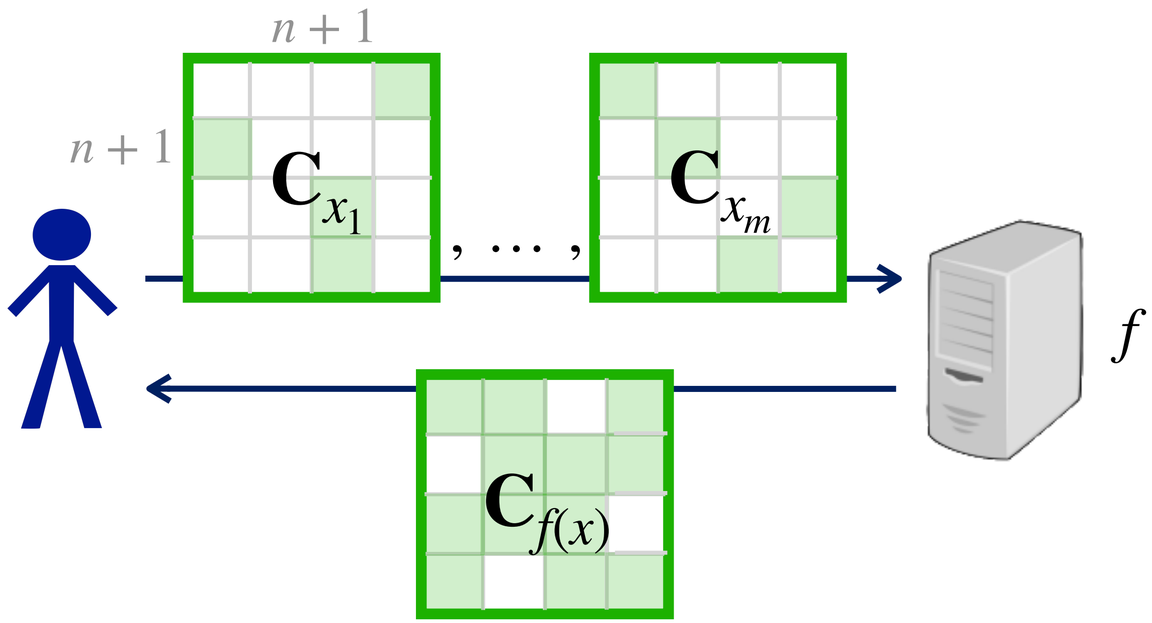
The key equation [GSW13]

Take sk to be a random vector $\mathbf{s} \in \mathbb{F}_q^n$.

Encrypt $x \in \mathbb{F}_q$ into a sparse matrix $\mathbf{C}_x \in \mathbb{F}_q^{(n+1)\times(n+1)}$, of which x is a "noisy" eigenvalue with sparse errors $\mathbf{e}$:

$$\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$$

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

The key equation [GSW13]

Take sk to be a random vector $\mathbf{s} \in \mathbb{F}_q^n$.

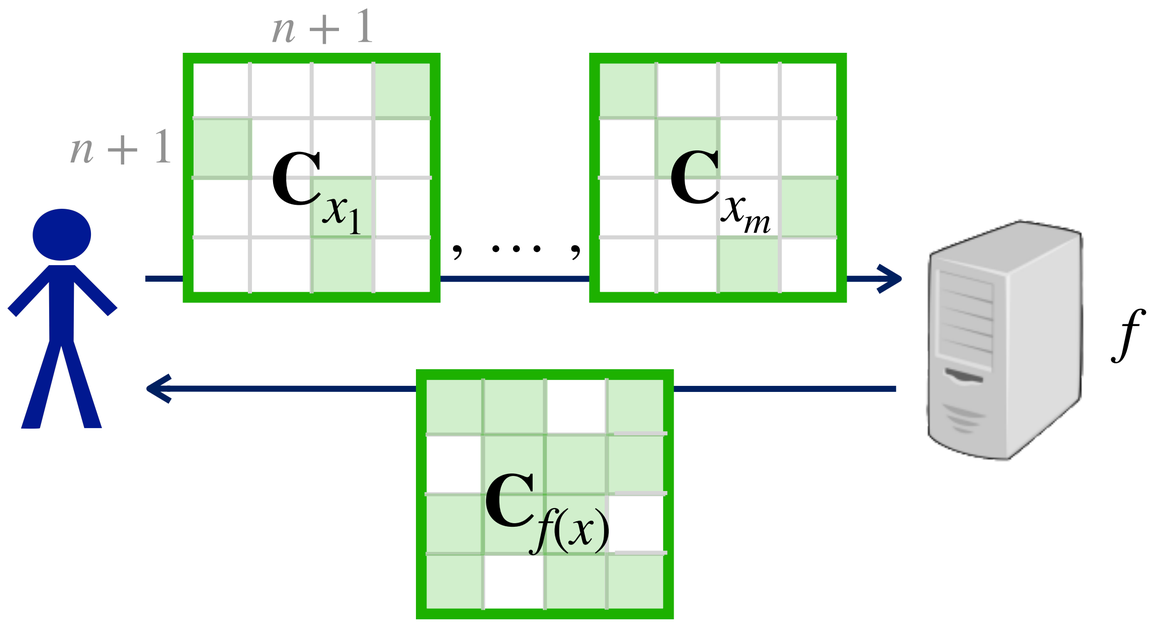
Encrypt $x \in \mathbb{F}_q$ into a sparse matrix $\mathbf{C}_x \in \mathbb{F}_q^{(n+1)\times(n+1)}$, of which x is a "noisy" eigenvalue with sparse errors $\mathbf{e}$:

$$\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$$

Add: $\mathbf{C}_x + \mathbf{C}_y$ is an encryption of $x + y$

Proof. $(\mathbf{C}_x + \mathbf{C}_y) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = (x+y) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + (\mathbf{e}_x + \mathbf{e}_y)$

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

The key equation [GSW13]

Take sk to be a random vector $\mathbf{s} \in \mathbb{F}_q^n$.

Encrypt $x \in \mathbb{F}_q$ into a sparse matrix $\mathbf{C}_x \in \mathbb{F}_q^{(n+1)\times(n+1)}$, of which x is a "noisy" eigenvalue with sparse errors $\mathbf{e}$:

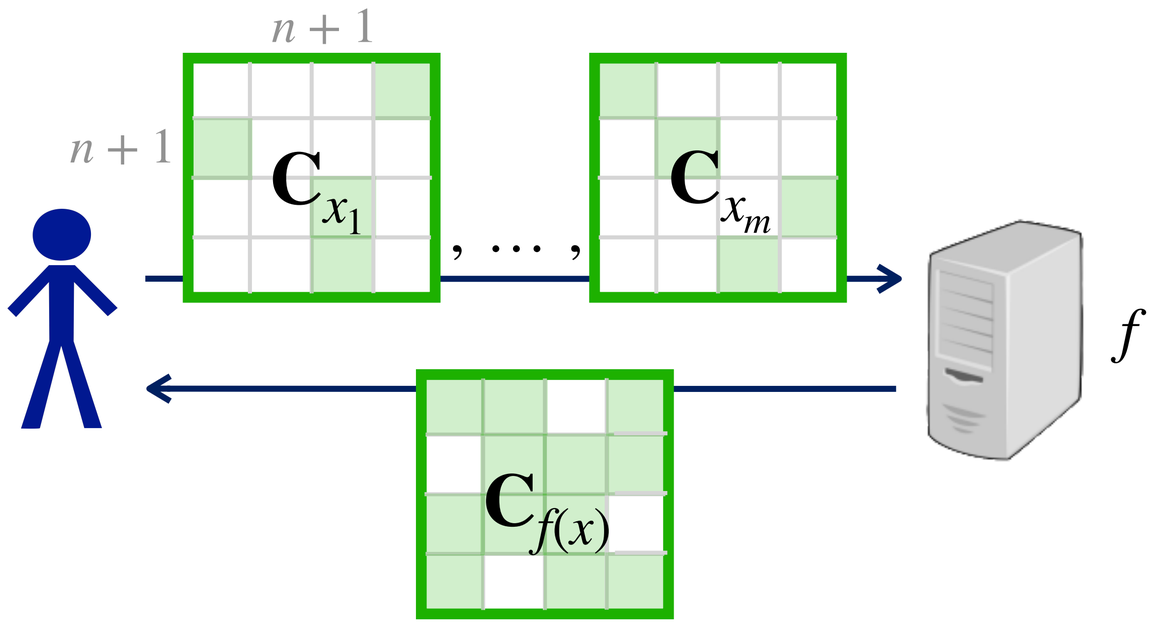
$$\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$$

Add: $\mathbf{C}_x + \mathbf{C}_y$ is an encryption of $x + y$

Mul: $\mathbf{C}_x \cdot \mathbf{C}_y$ is an encryption of $x \cdot y$

Proof. $(\mathbf{C}_x \cdot \mathbf{C}_y) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = (x \cdot y) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + (y \cdot \mathbf{e}_x + \mathbf{C}_x \cdot \mathbf{e}_y)$

The goal



- Homomorphic add is matrix addition
- Homomorphic mul is matrix mul
- Decryption is linear

The key equation [GSW13]

Take sk to be a random vector $\mathbf{s} \in \mathbb{F}_q^n$.

Encrypt $x \in \mathbb{F}_q$ into a sparse matrix $\mathbf{C}_x \in \mathbb{F}_q^{(n+1)\times(n+1)}$, of which x is a "noisy" eigenvalue with sparse errors $\mathbf{e}$:

$$\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$$

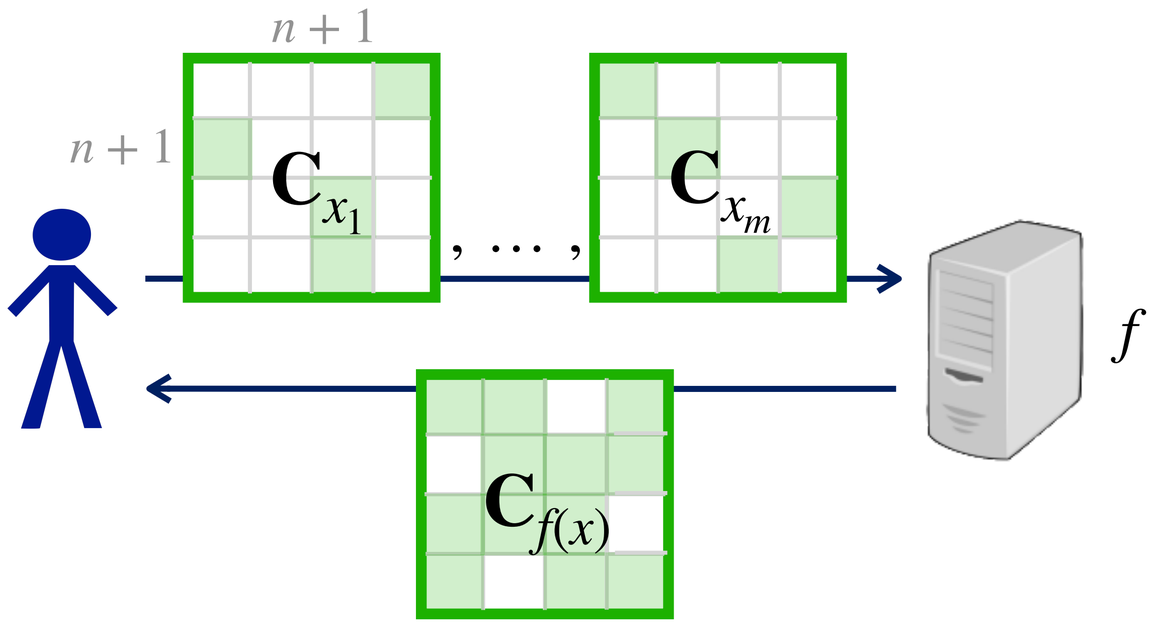
Add: $\mathbf{C}_x + \mathbf{C}_y$ is an encryption of $x + y$

Mul: $\mathbf{C}_x \cdot \mathbf{C}_y$ is an encryption of $x \cdot y$

Dec: output the last entry of $\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix}$

Proof. $\mathbf{C}_x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = x \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}$

The goal



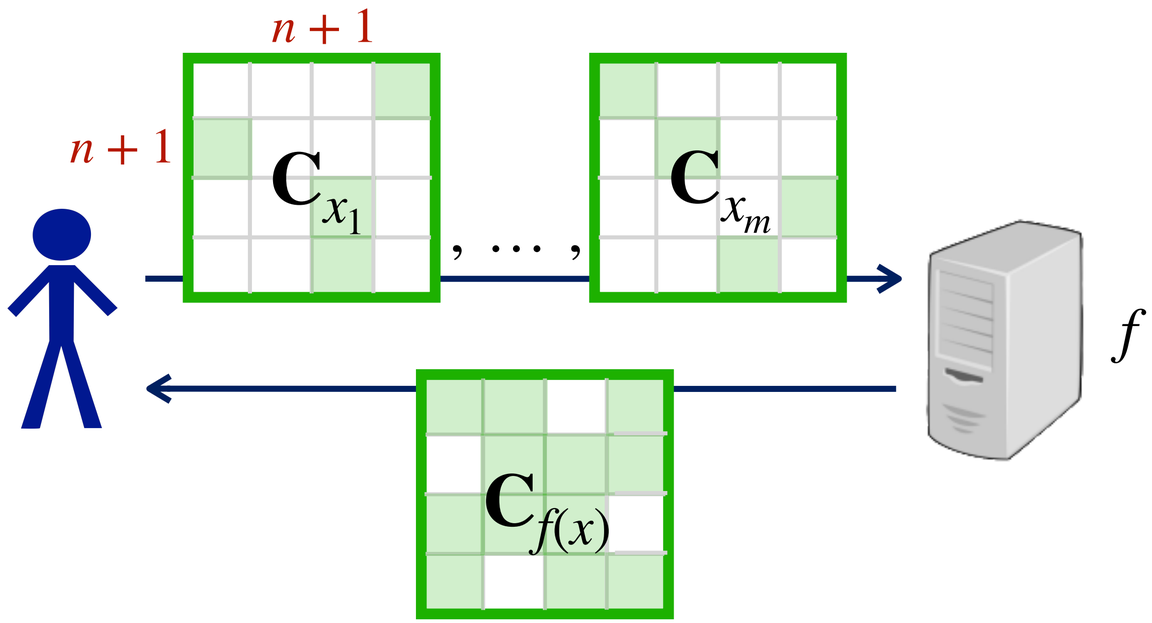
+ Correct, secure, and homomorphic.

On a secret key of dimension n, allows:

- $o(\log n)$ muls, followed by
- $n^{0.1-\epsilon}$ adds.

→ set n to match the desired number of operations

The goal



+ Correct, secure, and homomorphic.

On a secret key of dimension n, allows:

- $o(\log n)$ muls, followed by
- $n^{0.1-\epsilon}$ adds.

→ set n to match the desired number of operations

- Problem: Not compact!

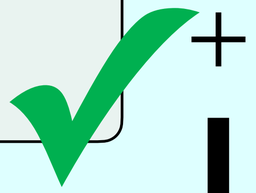
Due to limited homomorphism, $|f| = o(n^{0.1})$ bits

All ciphertexts are larger than $|f|$

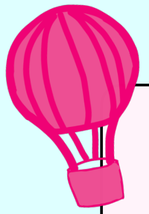
This talk

Sparse LPN

with modulus $q \geq 3$

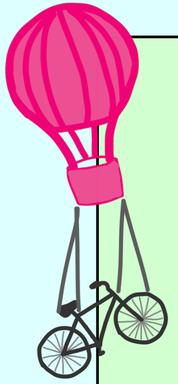


+



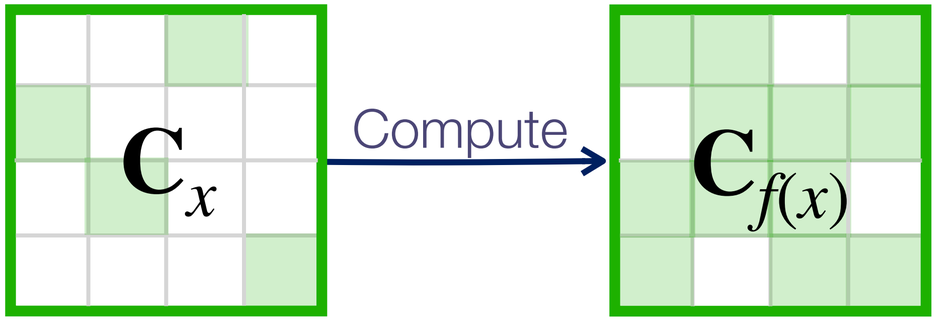
Linearly Homomorphic Encryption

over $\mathbb{F}_q$ (known from DDH/DCR)

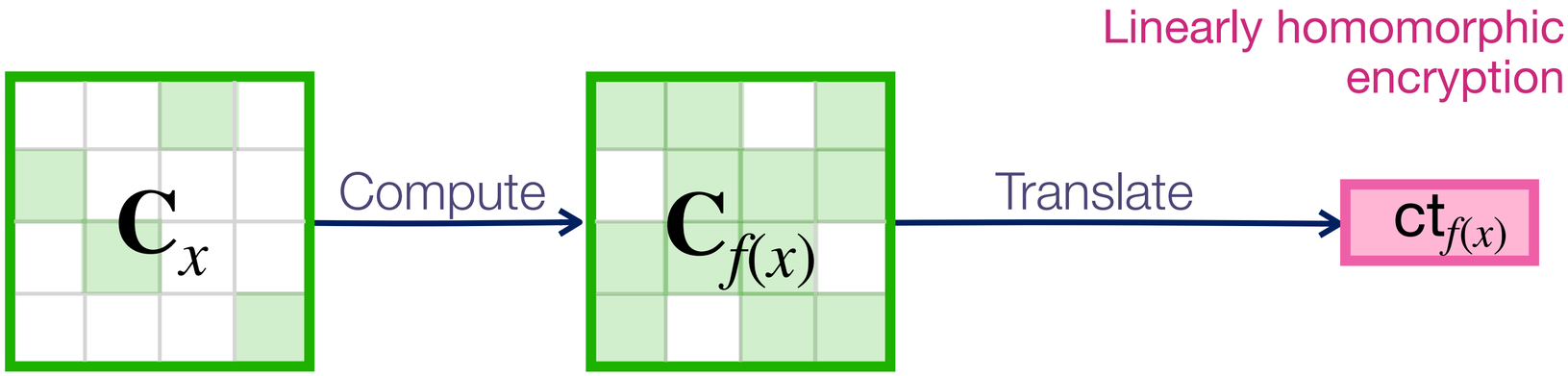


Somewhat Homomorphic Encryption that, for any $n = \text{poly}(\lambda)$ chosen during encryption, can perform $o(\log n)$ muls followed by n adds over $\mathbb{F}_q$

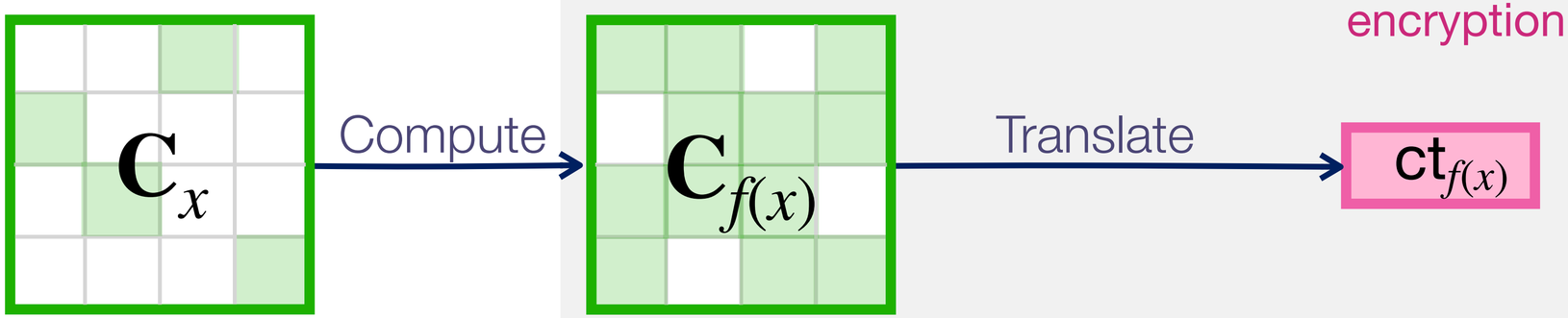
Fix: Use homomorphism to shrink the ciphertexts [G09]



Fix: Use homomorphism to shrink the ciphertexts [G09]



Fix: Use homomorphism to shrink the ciphertexts [G09]



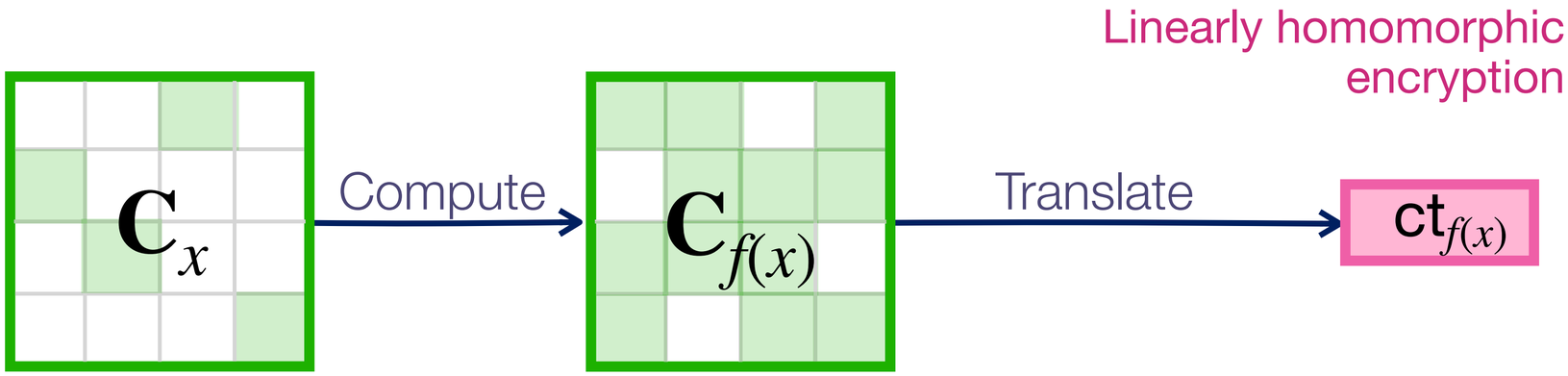
Linearly homomorphic encryption

Decrypting $\mathbf{C}_{f(x)}$ is a **linear** function f_{Dec} in sk

Plan:

1. Publish a **linearly** homomorphic enc. of sk
2. Anyone can homomorphically run f_{Dec} to get $\mathbf{ct}_{f(x)}$

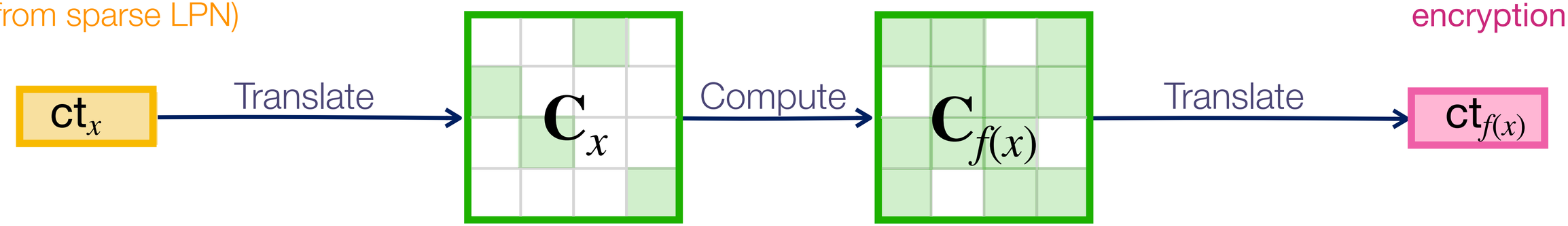
Fix: Use homomorphism to shrink the ciphertexts [G09]



Fix: Use homomorphism to shrink the ciphertexts [G09]

Encryption with cheap decrypt
(from sparse LPN)

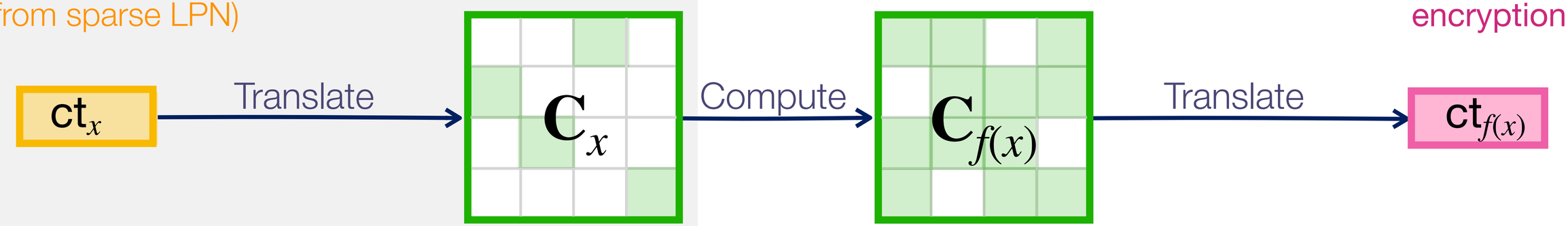
Linearly homomorphic encryption



Fix: Use homomorphism to shrink the ciphertexts [G09]

Encryption with cheap decrypt
(from sparse LPN)

Linearly homomorphic encryption

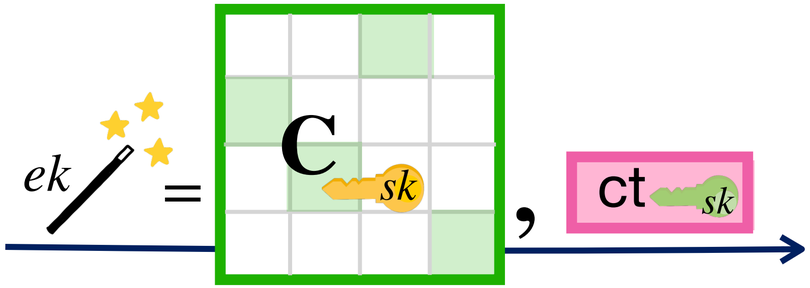


Decrypting ct_x is a **cheap** function f_{Dec} in sk

Plan:

1. Publish **somewhat**-homomorphic enc. of sk
2. Anyone can homomorphically run f_{Dec} to get $\mathbf{C}_x$

The final construction

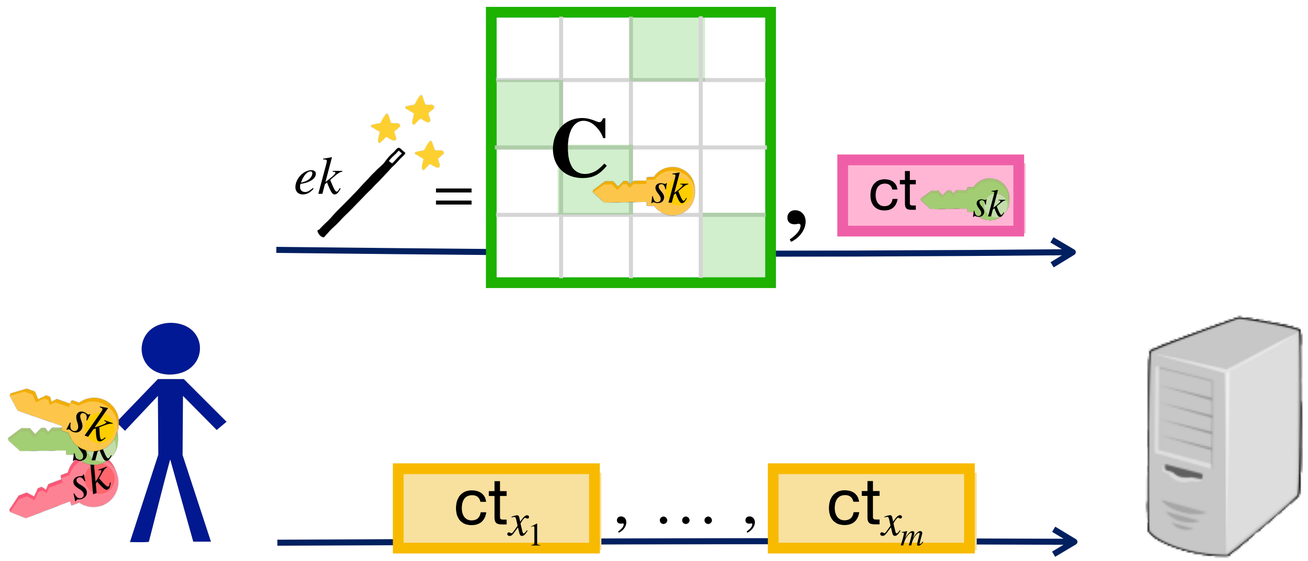


ct , *sk* : scheme 1 [has cheap decryption]

C , *sk* : scheme 2 [our sparse-LPN scheme]

ct , *sk* : scheme 3 [is linearly homomorphic]

The final construction

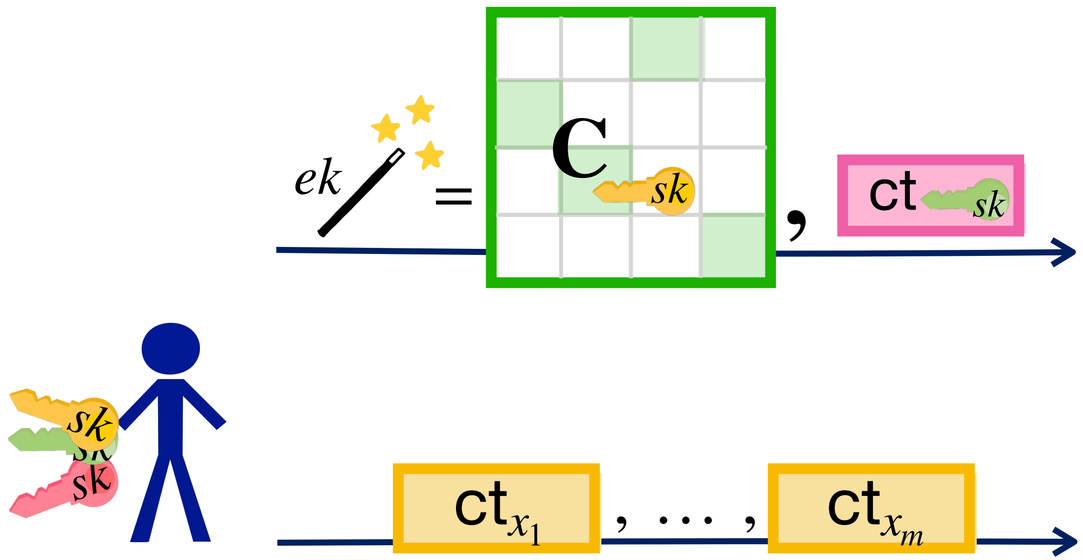


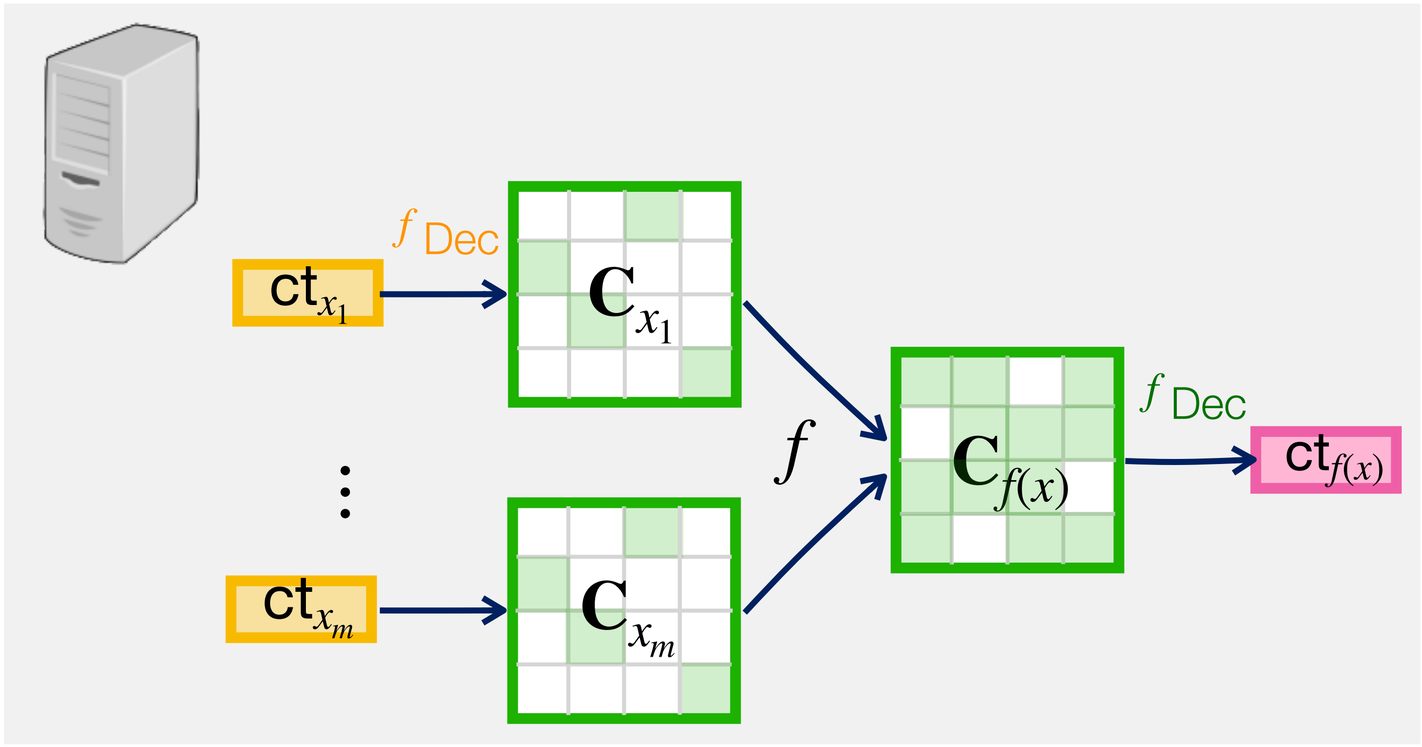
ct , *sk* : scheme 1 [has cheap decryption]

C , *sk* : scheme 2 [our sparse-LPN scheme]

ct , *sk* : scheme 3 [is linearly homomorphic]

The final construction



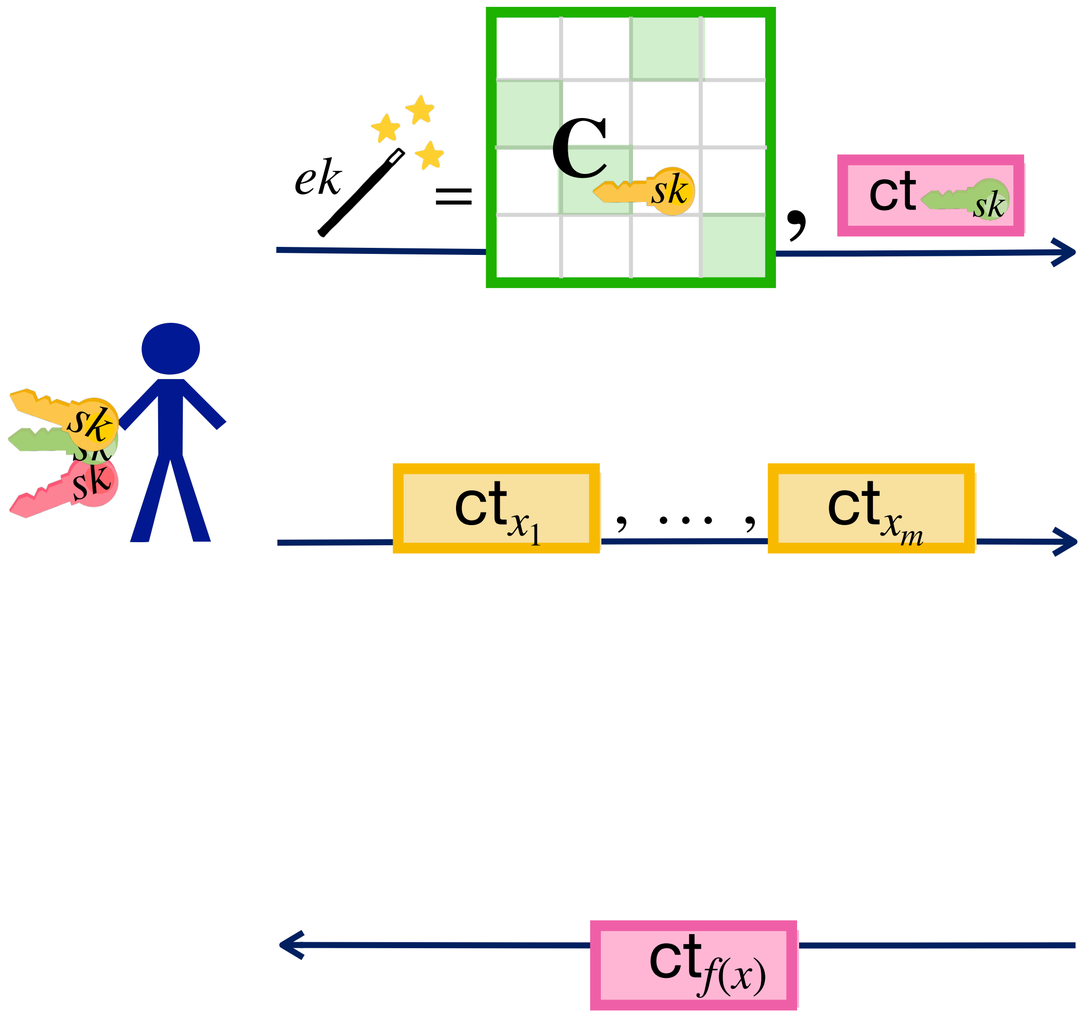


ct , sk : scheme 1 [has cheap decryption]

C , sk : scheme 2 [our sparse-LPN scheme]

ct , sk : scheme 3 [is linearly homomorphic]

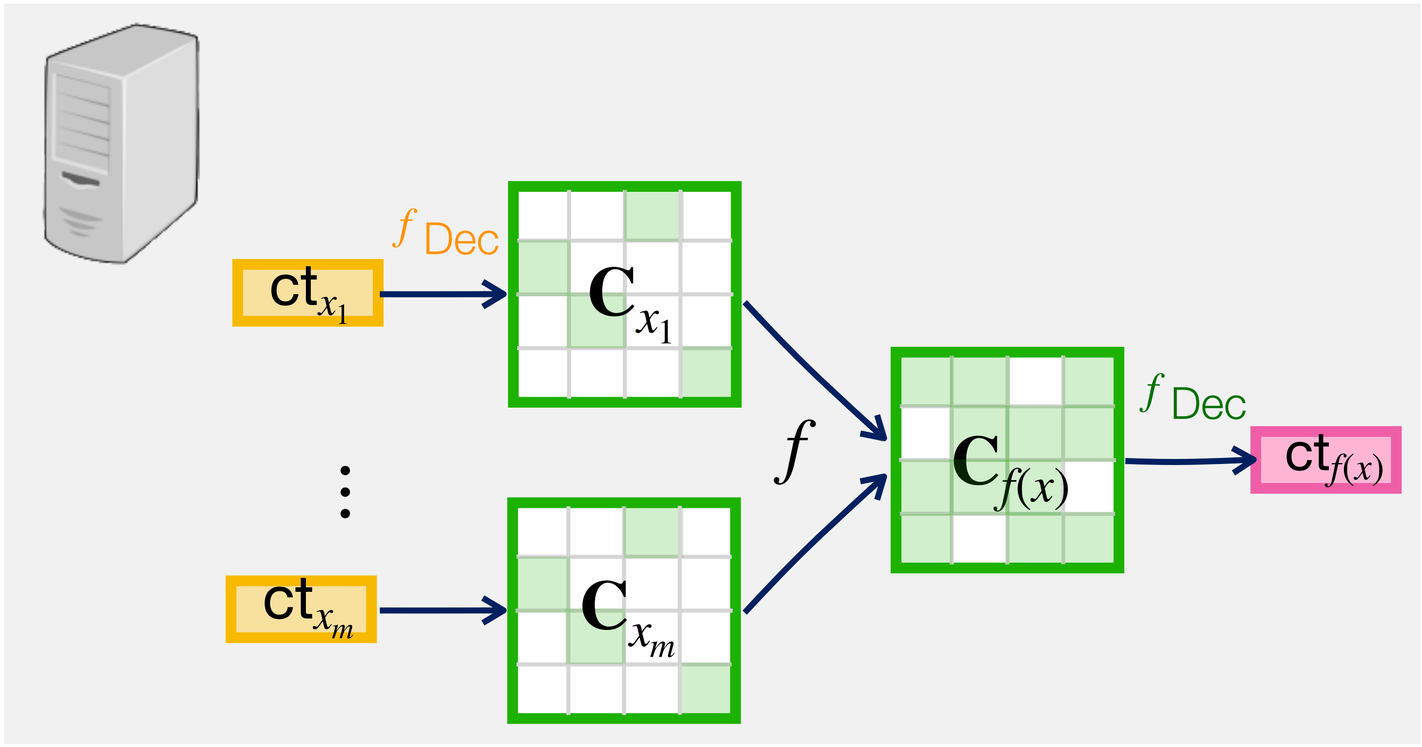
The final construction



ct , sk : scheme 1 [has cheap decryption]

C , sk : scheme 2 [our sparse-LPN scheme]

ct , sk : scheme 3 [is linearly homomorphic]

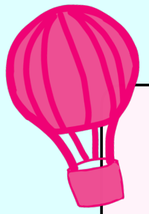


This talk

Sparse LPN

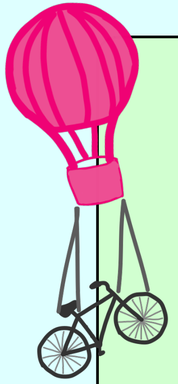
with modulus $q \geq 3$

+



Linearly Homomorphic Encryption

over $\mathbb{F}_q$ (known from DDH/DCR)

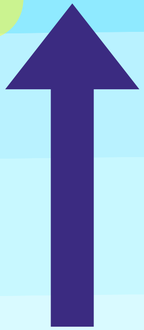


Somewhat Homomorphic Encryption that, for any $n = \text{poly}(\lambda)$ chosen during encryption, can perform $o(\log n)$ muls followed by n adds over $\mathbb{F}_q$

Fully Homomorphic Enc

Somewhat Homomorphic Enc

This work: can compute
- a few muls, followed by
- many adds
on encrypted data.

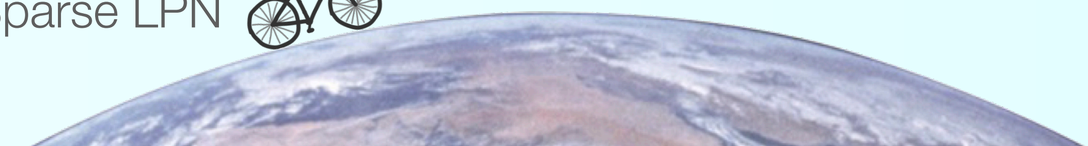


Linearly Homomorphic Enc

Number-theoretic
crypto

Private-Key Enc

Sparse LPN



Fully Homomorphic Enc

Somewhat Homomorphic Enc

This work: can compute
- a few muls, followed by
- many adds
on encrypted data.

Open: Can we shrink the evaluation key?
Can we get concrete efficiency?

Linearly Homomorphic Enc

Number-theoretic
crypto

Private-Key Enc

Sparse LPN 

Fully Homomorphic Enc

Somewhat Homomorphic Enc

This work: can compute
- a few muls, followed by
- many adds
on encrypted data.

Open: Can we build homomorphism from even fewer/weaker assumptions?

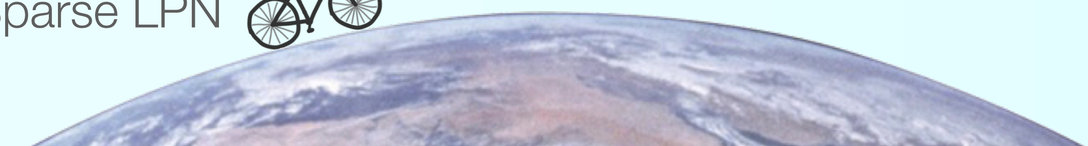
Open: Can we shrink the evaluation key? Can we get concrete efficiency?

Linearly Homomorphic Enc

Number-theoretic crypto

Private-Key Enc

Sparse LPN 



Fully Homomorphic Enc

???

Open: Can we bootstrap to full homomorphism?

Somewhat Homomorphic Enc

Open: Can we build homomorphism from even fewer/weaker assumptions?

This work: can compute
- a few muls, followed by
- many adds
on encrypted data.

Open: Can we shrink the evaluation key? Can we get concrete efficiency?

Linearly Homomorphic Enc

Number-theoretic crypto

Private-Key Enc

Sparse LPN

Fully Homomorphic Enc

???

Open: Can we bootstrap to full homomorphism?

Somewhat Homomorphic Enc

Open: Can we build homomorphism from even fewer/weaker assumptions?

This work: can compute
- a few muls, followed by
- many adds
on encrypted data.

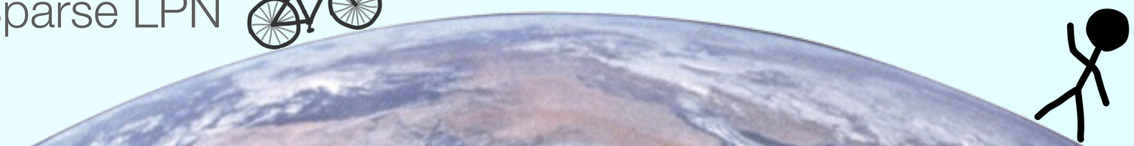
Open: Can we shrink the evaluation key? Can we get concrete efficiency?

Linearly Homomorphic Enc

Number-theoretic crypto

Private-Key Enc

Sparse LPN



Alexandra Henzinger
eprint.iacr.org/2024/1760
ahenz@csail.mit.edu

Thank you!