Cryptography as a Tool against White-Box Time-Bounded Adversaries

Ying Feng, Aayush Jain, David P. Woodruff

(CMU)

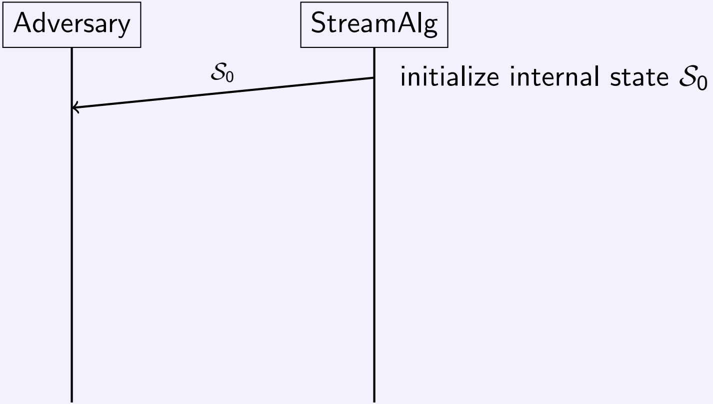
White-Box Adversarial Streaming Model

- The dataset is chosen adaptively by an adversary who sees the internal state of the algorithm
- Motivation: It captures richer adversarial scenarios such as
  - **Dynamic algorithms**: Maintain a dynamic data structure that correctly answers queries across sequentially arrived updates
    Dynamic model often considers an adaptive adversary that sees the entire data structure
  - **Machine learning**: Design robust models
    Many successful adversarial attacks use knowledge of internal algorithmic parameters and training weights

White-Box Adversarial Streaming Model

Definition

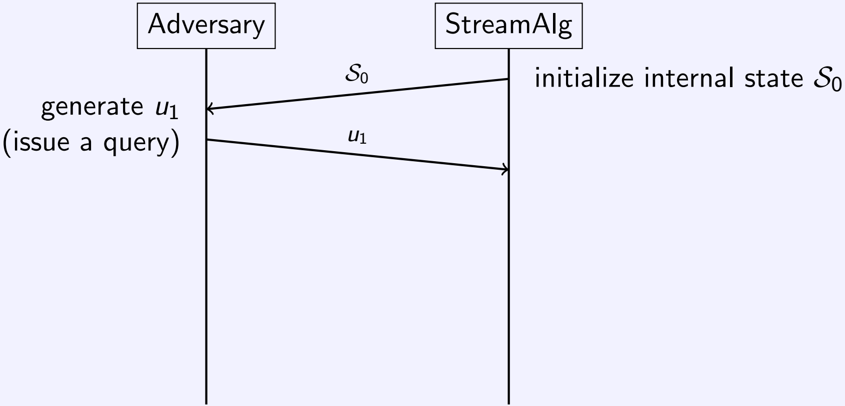
Two-player game between **StreamAlg** and **Adversary**:



White-Box Adversarial Streaming Model

Definition

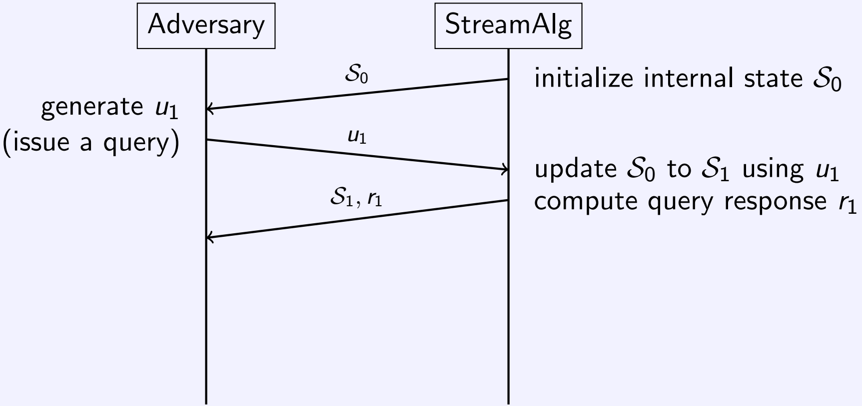
Two-player game between **StreamAlg** and **Adversary**:



White-Box Adversarial Streaming Model

Definition

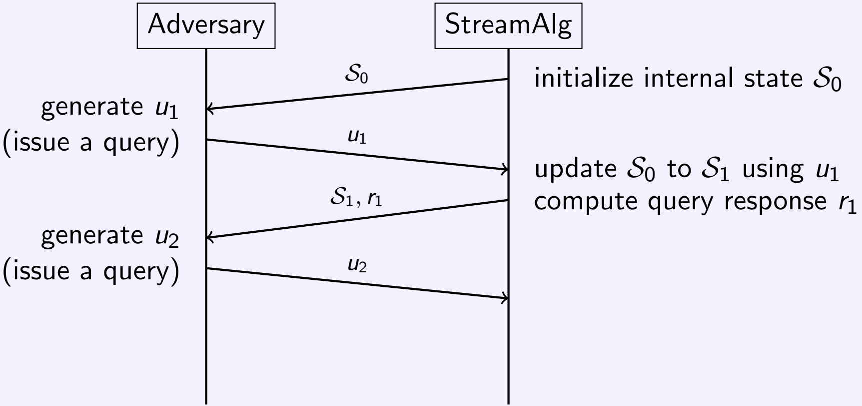
Two-player game between **StreamAlg** and **Adversary**:



White-Box Adversarial Streaming Model

Definition

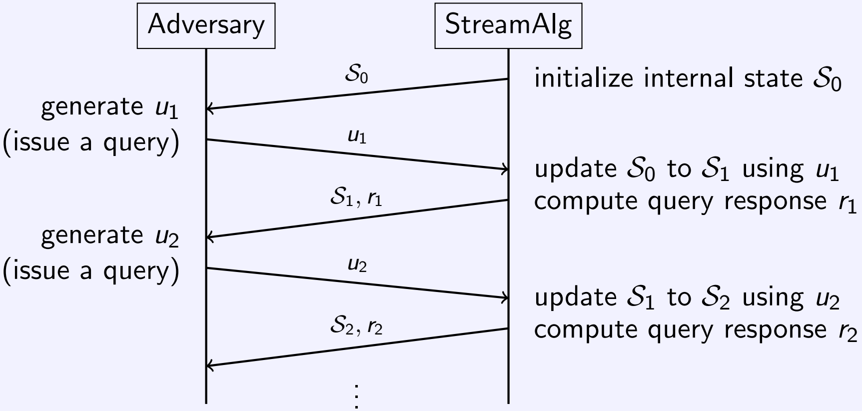
Two-player game between **StreamAlg** and **Adversary**:



White-Box Adversarial Streaming Model

Definition

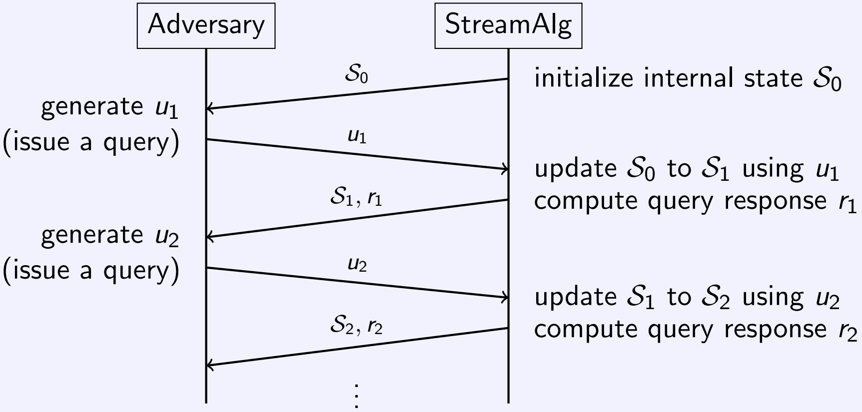
Two-player game between **StreamAlg** and **Adversary**:



White-Box Adversarial Streaming Model

Definition

Two-player game between **StreamAlg** and **Adversary**:



Adversary wins if r_t is incorrect at some time t

Main Problem

k-Sparse Recovery

Given an input vector x, if x contains at most k non-zero entries, recover x. Otherwise, report *invalid input*.

- Extensions to low-rank matrix and tensor recovery, Robust PCA, L_0 norm estimation

Main Problem

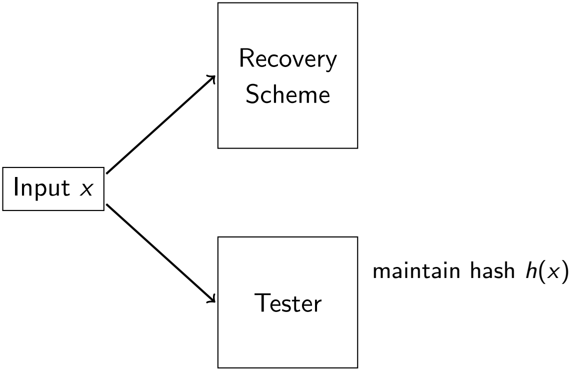
k-Sparse Recovery

Given an input vector x, if x contains at most k non-zero entries, recover x. Otherwise, report *invalid input*.

- Extensions to low-rank matrix and tensor recovery, Robust PCA, L_0 norm estimation
- Space lower bound of detecting k-sparsity, against white-box unbounded adversary: $\Omega(n)$.
  - ➢ consider time-bounded adversaries

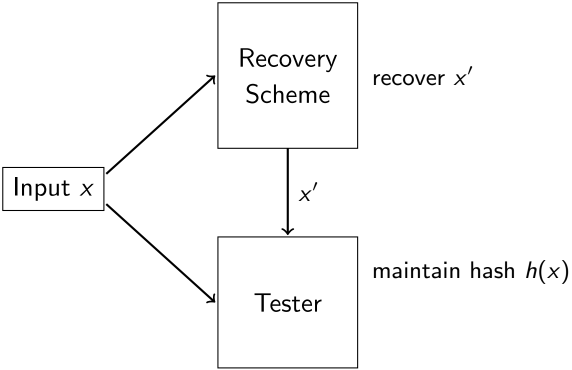
General Framework

- Existing deterministic k-sparse recovery scheme, plus
  - assumes that the input vector x is k-sparse, and only under this promise, recovers the k-sparse x
- A secure **tester** that verifies whether the input and the recovery output matches



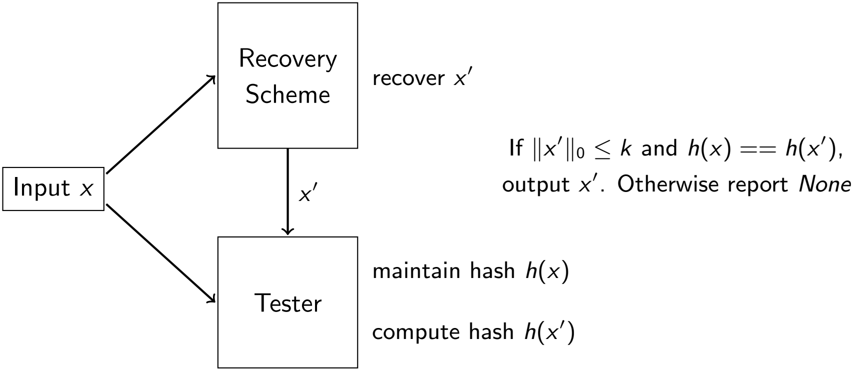
General Framework

- Existing deterministic k-sparse recovery scheme, plus
  - assumes that the input vector x is k-sparse, and only under this promise, recovers the k-sparse x
- A secure **tester** that verifies whether the input and the recovery output matches



General Framework

- Existing deterministic k-sparse recovery scheme, plus
  - assumes that the input vector x is k-sparse, and only under this promise, recovers the k-sparse x
- A secure **tester** that verifies whether the input and the recovery output matches



General Framework

- Tester $\approx$ *Streaming Friendly Collision-Resistant Hash Functions*, satisfying:
  - Small-sized hash key
  - Updatable
  - White-box adversarially robust
  - Time and space efficient
- The same framework also works for low-rank matrix and tensor recovery and RPCA, by changing the deterministic recovery scheme

Using Short Integer Solution Problem

Definition (SIS)

Given a uniformly random matrix $A \in \mathbb{Z}_q^{r \times n}$, find a nonzero integer vector $x \in \mathbb{Z}^n$ such that $Ax = 0 \mod q$ and $\|x\|_\infty \leq \beta$.

- From lattice-based cryptography

Using Short Integer Solution Problem

Definition (SIS)

Given a uniformly random matrix $A \in \mathbb{Z}_q^{r \times n}$, find a nonzero integer vector $x \in \mathbb{Z}^n$ such that $Ax = 0 \mod q$ and $\|x\|_\infty \leq \beta$.

- From lattice-based cryptography

Given adversarially generated x, define hash of x as Ax,

Claim

Assuming subexponential hardness of SIS and $r \geq \tilde{O}(k)$:
If x' is k-sparse and $Ax = Ax'$, then $x' = x$

Using Short Integer Solution Problem

Definition (SIS)

Given a uniformly random matrix $A \in \mathbb{Z}_q^{r \times n}$, find a nonzero integer vector $x \in \mathbb{Z}^n$ such that $Ax = 0 \mod q$ and $\|x\|_\infty \le \beta$.

- From lattice-based cryptography

Given adversarially generated x, define hash of x as Ax,

Claim

Assuming subexponential hardness of SIS and $r \ge \tilde{O}(k)$:
If x' is k-sparse and $Ax = Ax'$, then $x' = x$

- Otherwise adversary can brute-force all k-sparse vectors to find x'
- $x - x'$ is a SIS solution
- Robust against subexp-time adversary

Removing Dependency on k

Claim

Assuming subexponential hardness of SIS and $r \geq \tilde{O}(k)$:
If x' is k-sparse and $Ax = Ax'$, then $x' = x$

- Otherwise adversary can brute-force all k-sparse vectors to find x'

Removing Dependency on k

Claim

Assuming subexponential hardness of SIS and $r \geq \tilde{O}(k)$:
If x' is k-sparse and $Ax = Ax'$, then $x' = x$

- Otherwise adversary can brute-force all k-sparse vectors to find x'

Efficient reduction:

Claim

If x' is outputted by a polytime recovery algorithm and $Ax = Ax'$, then $x' = x$

- Standard collision-resistance
- Robust against polytime adversary

Using Fully Homomorphic Encryption

- A is a uniformly random matrix $\rightarrow$ expensive to store
- Want to find a **pseudorandom** matrix, with similar collision-resistance property

Using Fully Homomorphic Encryption

- A is a uniformly random matrix $\rightarrow$ expensive to store
- Want to find a **pseudorandom** matrix, with similar collision-resistance property

Definition (FHE)

An encryption scheme that enables function evaluation on **ciphertexts**, while yielding a ciphertext that encrypts the result as if the function was evaluated on plaintexts

$$\begin{tikzcd}
\boxed{\mathsf{ct}} \arrow[r, "\mathsf{FHE.Eval}_{\mathsf{pk}}(f,\ \mathsf{ct})"] & \boxed{\hat{\mathsf{ct}}} \arrow[d, "\mathsf{FHE.Dec}_{\mathsf{sk}}(\cdot)"] \\
\boxed{\mathsf{msg}} \arrow[u, "\mathsf{FHE.Enc}_{\mathsf{sk}}(\cdot)"] \arrow[r, dashed, "f(\mathsf{msg})"] & \boxed{f(\mathsf{msg})}
\end{tikzcd}$$

Using Fully Homomorphic Encryption

- Store $\mathsf{ct}_1, \mathsf{ct}_2, \cdots, \mathsf{ct}_{\log n}$ as seeds: think of them as encrypting a $\log n$-bit message $\mathsf{msg} \in [n]$

Using Fully Homomorphic Encryption

- Store ct_1, ct_2, $\cdots$, $\mathsf{ct}_{\log n}$ as seeds: think of them as encrypting a $\log n$-bit message $\mathsf{msg} \in [n]$
- Derive n ciphertexts $\hat{\mathsf{ct}}_1, \hat{\mathsf{ct}}_2, \cdots, \hat{\mathsf{ct}}_n$ by evaluating n functions $f_1, f_2, \cdots, f_n$ on $(\mathsf{ct}_1, \mathsf{ct}_2, \cdots, \mathsf{ct}_{\log n})$
  - $\hat{\mathsf{ct}}_i =$ the i-th column of sketching matrix

    Can be generated on the fly

Using Fully Homomorphic Encryption

- Store ct_1, ct_2, $\cdots$, $\mathsf{ct}_{\log n}$ as seeds: think of them as encrypting a $\log n$-bit message $\mathsf{msg} \in [n]$
- Derive n ciphertexts $\hat{\mathsf{ct}}_1, \hat{\mathsf{ct}}_2, \cdots, \hat{\mathsf{ct}}_n$ by evaluating n functions $f_1, f_2, \cdots, f_n$ on $(\mathsf{ct}_1, \mathsf{ct}_2, \cdots, \mathsf{ct}_{\log n})$
  - $\hat{\mathsf{ct}}_i =$ the i-th column of sketching matrix

    Can be generated on the fly
  - $f_i(\mathsf{msg}) = \begin{cases} 1 & \text{if } \mathsf{msg} == i \\ 0 & \text{otherwise} \end{cases}$

Using Fully Homomorphic Encryption

- Store ct_1, ct_2, $\cdots$, $\mathrm{ct}_{\log n}$ as seeds: think of them as encrypting a $\log n$-bit message $\mathrm{msg} \in [n]$
- Derive n ciphertexts $\hat{\mathrm{ct}}_1, \hat{\mathrm{ct}}_2, \cdots, \hat{\mathrm{ct}}_n$ by evaluating n functions $f_1, f_2, \cdots, f_n$ on $(\mathrm{ct}_1, \mathrm{ct}_2, \cdots, \mathrm{ct}_{\log n})$
    - $\hat{\mathrm{ct}}_i =$ the i-th column of sketching matrix

      Can be generated on the fly
    - $f_i(\mathrm{msg}) = \begin{cases} 1 & \text{if } \mathrm{msg} == i \\ 0 & \text{otherwise} \end{cases}$
- Hash $h(x)$ is the matrix-vector product
    - what is this linear combination of ciphertexts?

FHE with Additional Properties

What is this linear combination of ciphertexts?

Additional Property 1. (Linear Homomorphism)

If $\hat{\mathrm{ct}}_i$ encrypts a bit $b \in \{0,1\}$ and x_i is a small-normed integer, then there is a decryption algorithm that uniquely decodes $\sum_{i\in[n]} \hat{\mathrm{ct}}_i x_i$ to $\sum_{i\in[n]} b_i x_i$

- Satisfied by most known FHE schemes

FHE with Additional Properties

What is this linear combination of ciphertexts?

Additional Property 1. (Linear Homomorphism)

If $\hat{\mathrm{ct}}_i$ encrypts a bit $b \in \{0,1\}$ and x_i is a small-normed integer, then there is a decryption algorithm that uniquely decodes $\sum_{i\in[n]} \hat{\mathrm{ct}}_i x_i$ to $\sum_{i\in[n]} b_i x_i$

- Satisfied by most known FHE schemes
- $h(x) = h(x') \Rightarrow \sum_{i\in[n]} b_i x_i = \sum_{i\in[n]} b_i x'_i$

FHE with Additional Properties

What is this linear combination of ciphertexts?

Additional Property 1. (Linear Homomorphism)

If $\hat{\mathrm{ct}}_i$ encrypts a bit $b \in \{0,1\}$ and x_i is a small-normed integer, then there is a decryption algorithm that uniquely decodes $\sum_{i\in[n]} \hat{\mathrm{ct}}_i x_i$ to $\sum_{i\in[n]} b_i x_i$

- Satisfied by most known FHE schemes
- $h(x) = h(x') \Rightarrow \sum_{i\in[n]} b_i x_i = \sum_{i\in[n]} b_i x_i'$
- $b_i = f_i(\mathrm{msg}) = 1$ iff $i = \mathrm{msg}$,
  so $h(x) = h(x') \Rightarrow x_{\mathrm{msg}} = x'_{\mathrm{msg}}$

FHE with Additional Properties

What is this linear combination of ciphertexts?

Additional Property 1. (Linear Homomorphism)

If $\hat{\mathsf{ct}}_i$ encrypts a bit $b \in \{0, 1\}$ and x_i is a small-normed integer, then there is a decryption algorithm that uniquely decodes $\sum_{i \in [n]} \hat{\mathsf{ct}}_i x_i$ to $\sum_{i \in [n]} b_i x_i$

- Satisfied by most known FHE schemes
- $h(x) = h(x') \Rightarrow \sum_{i \in [n]} b_i x_i = \sum_{i \in [n]} b_i x'_i$
- $b_i = f_i(\mathsf{msg}) = 1$ iff $i = \mathsf{msg}$,
  so $h(x) = h(x') \Rightarrow x_{\mathsf{msg}} = x'_{\mathsf{msg}}$
- How to choose msg?
  - random $\mathsf{msg} \leftarrow [n]$

FHE with Additional Properties

What is this linear combination of ciphertexts?

Additional Property 1. (Linear Homomorphism)

If $\hat{\mathsf{ct}}_i$ encrypts a bit $b \in \{0, 1\}$ and x_i is a small-normed integer, then there is a decryption algorithm that uniquely decodes $\sum_{i \in [n]} \hat{\mathsf{ct}}_i x_i$ to $\sum_{i \in [n]} b_i x_i$

- Satisfied by most known FHE schemes
- $h(x) = h(x') \Rightarrow \sum_{i \in [n]} b_i x_i = \sum_{i \in [n]} b_i x'_i$
- $b_i = f_i(\mathsf{msg}) = 1$ iff $i = \mathsf{msg}$,
  so $h(x) = h(x') \Rightarrow x_{\mathsf{msg}} = x'_{\mathsf{msg}}$
- How to choose msg?
  - random $\mathsf{msg} \leftarrow [n]$
  - if msg is hidden from the adversary: $\Pr[x_{\mathsf{msg}} \neq x'_{\mathsf{msg}}] \geq \Pr[x \neq x']/n$

FHE with Additional Properties

Remaining question: Because all algorithm parameters (including sk) are revealed, we cannot securely encrypt/hide anything

FHE with Additional Properties

Remaining question: Because all algorithm parameters (including sk) are revealed, we cannot securely encrypt/hide anything

Additional Property 2. (Pseudorandomness)

The distribution of public keys and ciphertexts are indistinguishable, respectively, from some **truly random distributions** from the perspective of the adversary.

- Satisfied by most known FHE schemes

FHE with Additional Properties

Remaining question: Because all algorithm parameters (including sk) are revealed, we cannot securely encrypt/hide anything

Additional Property 2. (Pseudorandomness)

The distribution of public keys and ciphertexts are indistinguishable, respectively, from some **truly random distributions** from the perspective of the adversary.

- Satisfied by most known FHE schemes
- Allow us to sample truly random seeds $\tilde{\mathrm{ct}}_1, \tilde{\mathrm{ct}}_2, \cdots, \tilde{\mathrm{ct}}_{\log n}$ for the streaming algorithm
  - We switch to real ciphertexts $\mathrm{ct}_1, \mathrm{ct}_2, \cdots, \mathrm{ct}_{\log n}$ that encrypt msg only in the security proof
  - We can derive collision-resistant hash function using these random seeds, in the same way as using real ciphertexts

Results

Using the FHE scheme in [GSW], which is based on the hardness of the Learning-with-Error (LWE) problem:

- Assuming sub-exponential hardness of LWE, our construction takes $\operatorname{poly}\log(n)$ space and update time[a] in the stream.
- Assuming polynomial hardness of LWE, our construction takes $n^c \cdot \operatorname{poly}\log(n)$ space and update time, for an arbitrarily small constant $c > 0$.

[a] plus the space and time taken by the deterministic recovery scheme

White-Box Adversarial Distributed Model

We also consider a related distributed model:

- A message-passing coordinator model with t servers and one central coordinator
- Each server receives a white-box adversarially generated data vector x_i and communicates with the coordinator
- **Goal**: design a communication protocol that allows the coordinator to compute some function on the aggregated data $\sum_{i\in[t]} x_i$

White-Box Adversarial Distributed Model

We also consider a related distributed model:

- A message-passing coordinator model with t servers and one central coordinator
- Each server receives a white-box adversarially generated data vector x_i and communicates with the coordinator
- **Goal**: design a communication protocol that allows the coordinator to compute some function on the aggregated data $\sum_{i \in [t]} x_i$

Use a strong definition of white-box adversary:

- Before the communication starts, the adversary observes the complete random string that is going to be used by all parties
- It then generates data for each server

Results

We can adapt our streaming algorithm to construct a communication protocol for the distributed model. Using the LWE-based FHE in [GSW]:

- Bits of communication $\approx$ space of streaming algorithm
- Process time $\approx n \cdot$ stream update time

Results

We can adapt our streaming algorithm to construct a communication protocol for the distributed model. Using the LWE-based FHE in [GSW]:

- Bits of communication $\approx$ space of streaming algorithm
- Process time $\approx n \cdot$ stream update time

Using a FHE scheme in [BV], which is based on a more structured Ring Learning-with-Error problem:

- Assuming **polynomial** hardness of Ring-LWE, our protocol takes $n \cdot \text{poly}\log(n)$ process time (as opposed to $n^{1+c} \cdot \text{poly}\log(n)$ assuming polynomial LWE)

Results

We can adapt our streaming algorithm to construct a communication protocol for the distributed model. Using the LWE-based FHE in [GSW]:

- Bits of communication $\approx$ space of streaming algorithm
- Process time $\approx n \cdot$ stream update time

Using a FHE scheme in [BV], which is based on a more structured Ring Learning-with-Error problem:

- Assuming **polynomial** hardness of Ring-LWE, our protocol takes $n \cdot \operatorname{poly}\log(n)$ process time (as opposed to $n^{1+c} \cdot \operatorname{poly}\log(n)$ assuming polynomial LWE)

- By packing partitioned segments of the input vector as ring elements
- Efficiently operations on ring elements using FFT