Compact Storage for Homomorphic Encryption

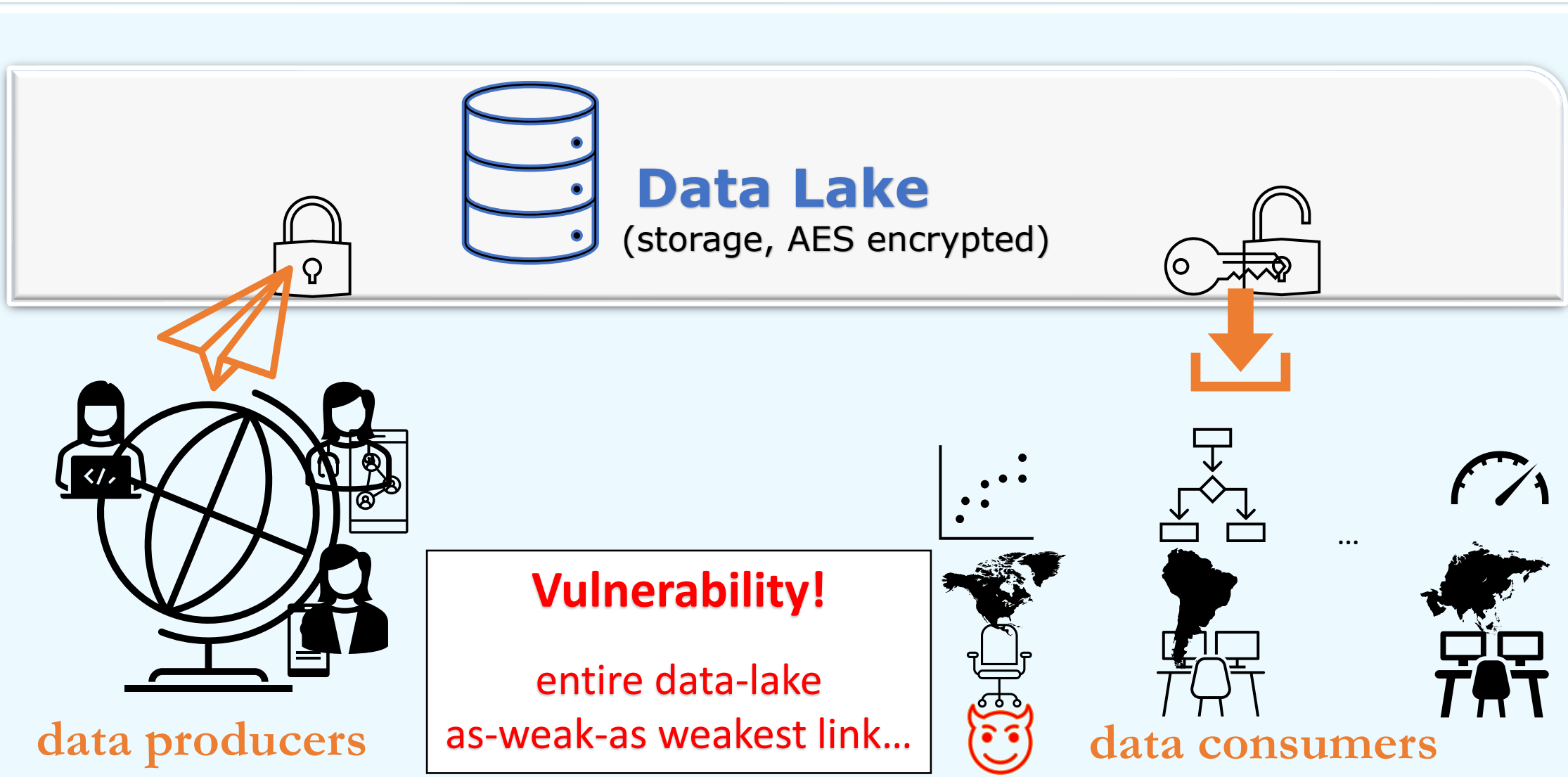
Adi Akavia & Neta Oren

Boaz Sapir & **Margarita Vald**

University of Haifa

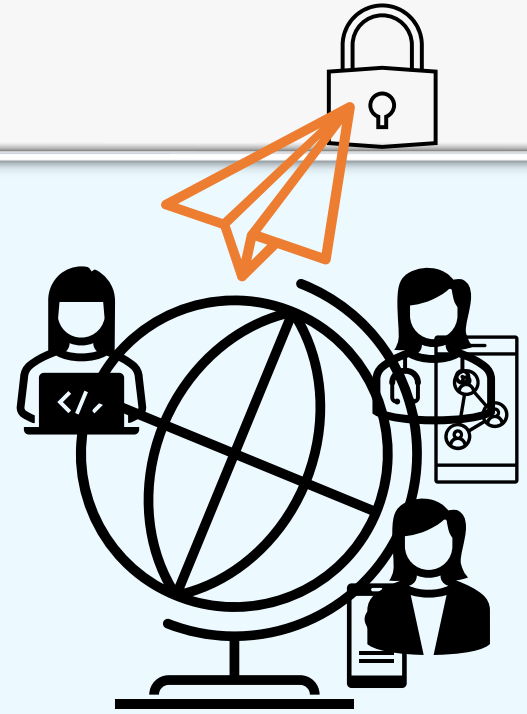
Intuit, Israel Ltd.

Enterprise Architecture & Threats



Enhancing Privacy using HE

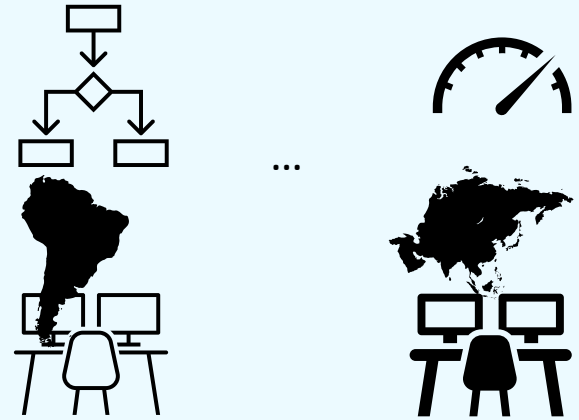
*Only authorized entities can decrypt



data producers

Data Lake
(storage, encrypted with HE)

adversary sees only
encrypted data –
secrecy is provided!



data consumers

The HE approach is simple and appealing,
but is it ready for use in practice?

Key Complexity Bottlenecks in HE

☒ **Time** significant runtime overheads

But: Much recent progress!

Today's focus

☒ **Storage** 10×-10,000× overhead over AES

Prior Works I: Store **AES** ciphertext, **Transform** to **HE** via **homomorphic decryption**

[Naehrig etal'11]

Implementations for AES [Gentry-Halevi-Smart'12, Doroz etal'14…]

& for tailored ciphers [LowMC, Kreyvium, FLIP, RASTA,MASTA, HERA…]

☒ despite much progress, still **too slow** for retrieval at scale

Key Complexity Bottlenecks in HE

☒ **Time** significant runtime overheads

But: Much recent progress!

Today's focus

☒ **Storage** 10×-10,000× overhead over AES

Prior Works II – Rate-1 HE [Gentry-Halevi'19, Brakerski etal'19]**:**
via **packing & compressing** many data items in each ciphertext

Issues:
- ☒ Compressed ciphertexts only support **additive homomorphism**
- ☒ Uncompressing is **slow**
- ☒ Packing determined at storage => **no "cherry picking"** of data items to retrieve

Our Result: Compact Storage for HE

Our approach:

Store a **secret share**, at retrieval securely transform to **HE** (e.g., CKKS) in 2-server model

Achieving:

- ☑ **Rate-1 storage (no storage overhead)**
- ☑ **Data privacy**
- ☑ **Unrestricted homomorphism**
- ☑ **Fast runtime** ~2X comp. storing & retrieving HE ciphertexts
- ☑ **Dynamic control** at retrieval of
  - data cherry picking
  - HE scheme & params
  - packing profile

New Tool: **Secret Sharing** with Homomorphic Reconstruction over Reals

- ☑ **Rate-1** shares
- ☑ Reconstruction requires only **additive homomorphism** over the **reals** (**no** **modular** reduction!)
- ☑ **Fast** to reconstruct over data encrypted with **CKKS**

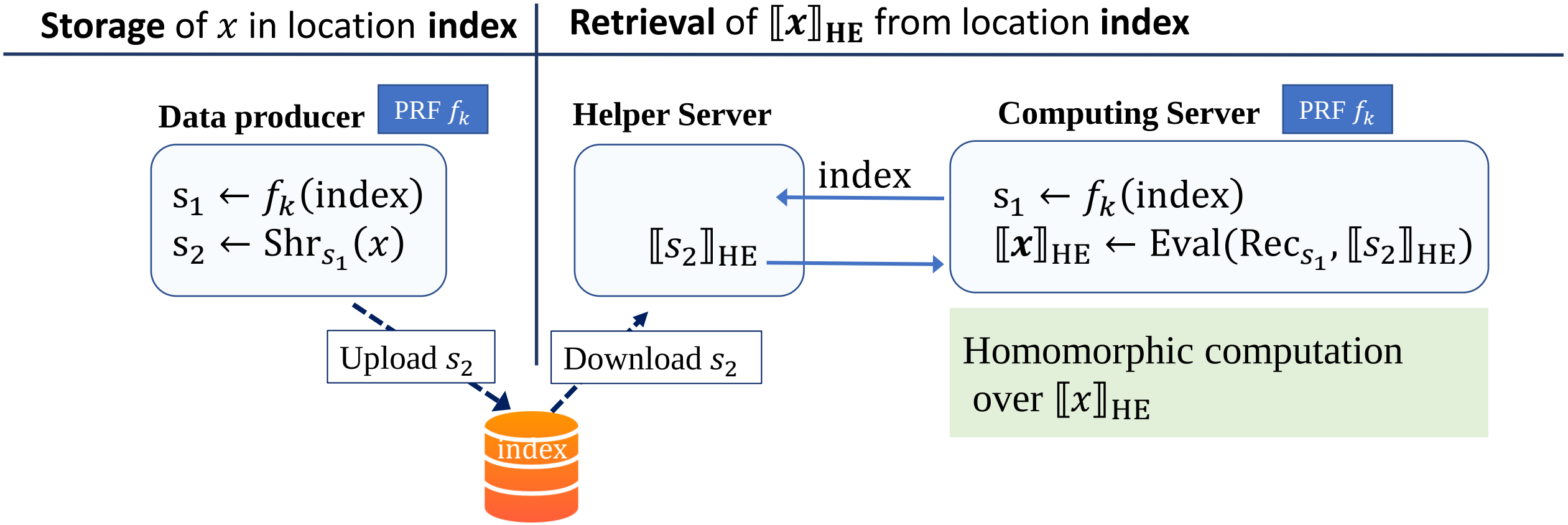
Prior perfect secret sharing – **modular** reduction required in Share & Rec

Slow when plaintext arithmetic is over **reals** as in **CKKS**

Our Compact Storage Construction

Generic Compact Storage for HE

Using PRFs and 2-out-2 Secret sharing with
random 1st share & linear homomorphic reconstruct



Empirical Evaluation

Our System

Instantiated with:

- Our **secret sharing** scheme for **reals**
- **CKKS** scheme in Microsoft SEAL v3.6.2

Deployed on:

- **AWS EC2** with **S3 storage** and **Google Cloud**

Empirical Evaluation:

Storage Size & Runtime

Our storage: 10× to 10^4× better than the baseline

Our runtime (amortized): $10\mu s$

Our cost of storage & retrieval:
outperforms the baseline
for $\leq 10^{16}$ retrievals/month with 25PB storage

*Baseline: Storing & retrieving HE ciphertexts

HE-Retrieve then Homomorphic Eval Decision-Tree

Baseline w. opt. HE params & packing profile

	Versatile Homomorphic Computations?	Cherry Picking Data?	Storage per Sample (KB)	Runtime per Sample (ms)
A	×	×	0.50	**0.53**
B	✓	×	0.50	1.70
C	×	✓	3600.00	74.23
D	✓	✓	7200.00	158.39
Ours	✓	✓	**0.05**	0.57

Our **Storage:** 10× to 10^4× better than baseline

Runtime: 1.07× **best** baseline

HE-retrieval only runtime: $9\mu s$ ~1.57% of retrieve-then-eval runtime

Summary

1) Compact storage with privacy preserving HE-retrieval in 2-server model

- **Rate-1**
- **Fast** runtime, nearly as fast as directly storing and retrieving HE ciphertexts
- **Dynamic control**, at retrieval time, of Retrieved data items,
  HE parameters,
  Packing profile,

2) Secret sharing w. **additive homomorphic** reconstruction over **reals**

3) Implementation using **AWS EC2, S3 bucket and Google Cloud**.


THANK
YOU