Compact Storage for Homomorphic Encryption

Adi Akavia & Neta Oren  Boaz Sapir & Margarita Vald

University of Haifa  Intuit, Israel Ltd.
Enterprise Architecture & Threats

Data Lake
(storage, AES encrypted)

Vulnerability!
entire data-lake
as-weak-as weakest link...
Enhancing Privacy using HE

Data Lake
(storage, encrypted with HE)

adversary sees only encrypted data – secrecy is provided!

*Only authorized entities can decrypt
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!

- **Storage**: $10 \times -10,000 \times$ overhead over AES

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

- Implementations for AES
- & for tailored ciphers

- Despite much progress, still too slow for retrieval at scale
Key Complexity Bottlenecks in HE

- **Time**: significant runtime overheads
  - But: Much recent progress!

- **Storage**: $10 \times - 10,000 \times$ overhead over AES

Prior Works II – **Rate-1 HE** [Gentry-Halevi’19, Brakerski et al’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 \(\sim 2\times\) 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

Storage of $x$ in location $\text{index}$

Data producer $\text{PRF } f_k$

$s_1 \leftarrow f_k(\text{index})$

$s_2 \leftarrow \text{Shr}_{s_1}(x)$

Helper Server

Helper Server

$s_2 \leftarrow \text{Eval}(\text{Rec}_{s_1}, [s_2]_{\text{HE}})$

$s_1 \leftarrow f_k(\text{index})$

$[x]_{\text{HE}} \leftarrow \text{Eval}(\text{Rec}_{s_1}, [s_2]_{\text{HE}})$

Computing Server $\text{PRF } f_k$

Download $s_2$

Homomorphic computation over $[x]_{\text{HE}}$

Storage of $x$ in location $\text{index}$

Retrieval of $[x]_{\text{HE}}$ from location $\text{index}$
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: $10 \times$ to $10^4 \times$ 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

<table>
<thead>
<tr>
<th></th>
<th>Versatile Homomorphic Computations?</th>
<th>Cherry Picking Data?</th>
<th>Storage per Sample (KB)</th>
<th>Runtime per Sample (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>×</td>
<td>×</td>
<td>0.50</td>
<td>0.53</td>
</tr>
<tr>
<td>B</td>
<td>✓</td>
<td>×</td>
<td>0.50</td>
<td>1.70</td>
</tr>
<tr>
<td>C</td>
<td>×</td>
<td>✓</td>
<td>3600.00</td>
<td>74.23</td>
</tr>
<tr>
<td>D</td>
<td>✓</td>
<td>✓</td>
<td>7200.00</td>
<td>158.39</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>✓</td>
<td>0.05</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Baseline w. opt. HE params & packing profile

Our Storage: $10\times$ to $10^4\times$ better than baseline

Runtime: $1.07\times$ 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