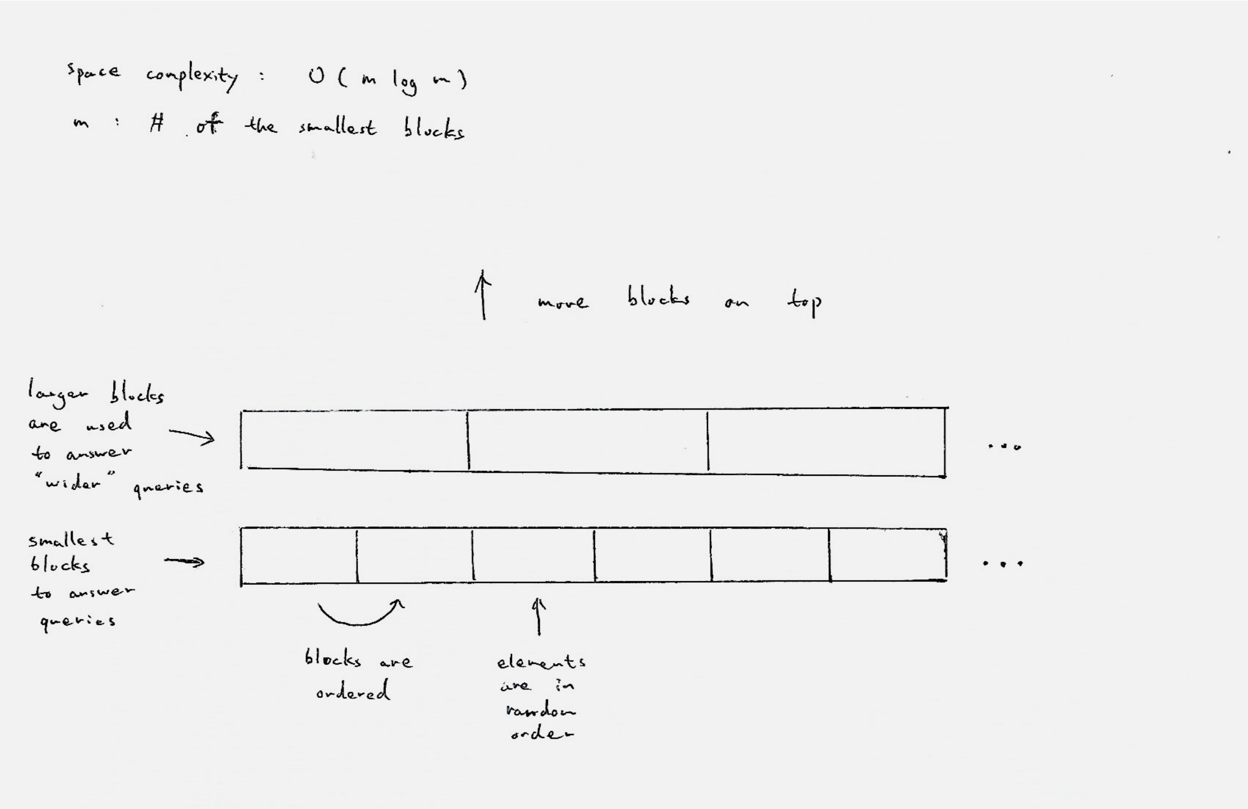
1 Papers referred to

1. Marie-Sarah Lacharite, Brice Minaud. **Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage**
2. **K-Nearest-Neighbour Search on Encrypted Data** (Not published yet)
3. Rafail Ostrovsky. **Software Protection and Simulation on Oblivious RAMs**
4. Data structure and algorithms I have learnt during undergraduate.

2 A new OPE scheme

- The idea is to use 'searchable' data structures to pre-process the queries so that the queries do not leak the sets used in 1.
- Sketch of the scheme:



space complexity : O (m log m)
m : # of the smallest blocks
↑ more blocks on top
larger blocks are used to answer "wider" queries
smallest blocks to answer queries
...
...
blocks are ordered
elements are in random order

- Instead of lookup table to identify the block to retrieve, one can use PRF on the endpoints (in terms of range covered by the block) of the block.
- Claim: the block returned contains at most twice number of items than the actual items.
- Problems with the scheme:
  - The claim is not quite true: consider the range spanning the second and third block in the bottom layer, there is no single block to return from the layer on top. However, there is a simple fix to this, and does not change space complexity (**To be discussed in the next meeting**).

  - If the range is 1 to 2^k then this scheme is not ideal in terms of space complexity. I claim that the scheme can be modified to achieve a very low overhead (**To be discussed in the next meeting**).

- To think: what does the scheme leak?

3 ORAM

- Confirms that the lower-bound of overhead is indeed $\mathcal{O}(\log n)$.
- Question to be answered: is ORAM any similar to OPE in the sense that, in order to hide access pattern, how much overhead does it cost?

4 Syntax of encrypted database

- We can think of database in three levels: a string of zeros and ones in the most abstract form, a database as a set of strings, and a database with all the details.
- We will formalize syntax on the most abstract form first.
- Notation (need better names):
  - **Initialise**: $1^\lambda \to \text{params}^*$
  - **KeyGen**: $1^\lambda \to \mathbf{sk}$
  - **Enc**: $1^\lambda \times \mathbf{sk} \times \mathbf{DB} \to \mathbf{EDB}$
  - **Dec**: $1^\lambda \times \mathbf{sk} \times \mathbf{EDB} \to \mathbf{DB}$
  - $\mathcal{Q}$: $\mathbf{DB} \times Q \to \mathbf{DB}$
  - **encQ**: $1^\lambda \times \mathbf{sk} \times Q \to \mathbf{eQ} \times \mathbf{iQ}$
  - **EQ**: $1^\lambda \times \mathbf{sk} \times \mathbf{EDB} \times \mathbf{eQ} \to \mathbf{EDB}$
  - **DQ**: $1^\lambda \times \mathbf{sk} \times \mathbf{EDB} \times \mathbf{iQ} \to \mathbf{DB}$
- Additional notation:
  - Π_i: projection of i-th component of the arguments
  - id: identity function
- Commutative diagram:

$$\begin{tikzcd}
\mathbf{DB} \times \mathbf{Q} \arrow[r, "\Pi_1"] \arrow[dddd, "id"'] \arrow[ddr, "\Pi_2"] & \mathbf{DB} \arrow[r, bend left, "Enc"] & \mathbf{EDB} \arrow[l, bend left, "Dec"] \arrow[rr, "\mathbf{EQ}"] & & \mathbf{eR} \arrow[dddd, "\mathbf{DQ}"] \\
& & & \mathbf{eQ} \arrow[ur, "\mathbf{EQ}"] & \\
& \mathbf{Q} \arrow[r, "\mathcal{Q}"] & \mathbf{eQ} \times \mathbf{iQ} \arrow[ur, "\Pi_1"] \arrow[dr, "\Pi_2"] & & \\
& & & \mathbf{iQ} \arrow[dr, "\mathbf{DQ}"] & \\
\mathbf{DB} \times \mathbf{Q} \arrow[rrrr, "\mathcal{Q}"] & & & & \mathbf{R}
\end{tikzcd}$$