Sublinear secure computation?

- All protocols thus far were circuit-based
  → any circuit for a non-trivial function has linear size

- Any secure computation protocol requires at least linear time
  → if a party does not touch every bit of input, this leaks information

Idea:

- Move to a setting where parties evaluate some function \( l \) times

- Hope for complexity \( O(n) + o(n^l) \)
  \( \Rightarrow \) amortized complexity \( O(n^l / l) + o(n) \)
  \( \Rightarrow \) \( l \) large \( \Rightarrow \) sublinear amortized complexity

- Or preprocessing (expensive)
  → online computations sublinear in \( n \)

Example: **Binary-Search**:

- Init setup
- For each binary-search computation
  - each lookup for the binary-search algorithm is compiled into a sequence of actual lookups in the ORAM.
One-sided secure protocol: (privacy of client)

Client

\[ \hat{D} \]

\[ \text{st}, \hat{D} \leftarrow \text{Init}(D) \]

\[ \hat{D} \]

Server \((D)\)

\[ \]

Two-sided secure protocol:

Client

\[ \text{st} \]

run ORAM-compiled
algorithm locally

whenever \( \hat{D}[i] \) is needed, send \( \hat{D} \) to
the server

\[ \hat{D}[i] \]

\[ \text{fetch} = \hat{D}[i] \]

\[ 16 (\text{op} = \text{write}) \]

\[ \hat{D}[i] = \text{data} \]

Server \((D)\)

\[ \]

\( \hat{D} \)

\[ \text{st} \]

\[ \text{Init} \]

\[ \text{st}_1, \hat{D}, \text{st}_2 \]

\( \text{st} = \text{st}_1 \otimes \text{st}_2 \)

\[ \]

Client \((i, \text{st}_1)\)

\[ \text{NextInstr} \]

\[ \text{st}_2 \]

\[ \text{st}_2, \text{GetInst} \]

\[ \text{op}, i, \text{data}, \text{st}_2 \]

\[ 16 (\text{op} = \text{write}) \]

\[ \hat{D}[i] = \text{data} \]

Server \((\hat{D}, \text{st}_2)\)

\[ \text{NextInstr} \]

\[ \text{st}_2 \]

\[ \text{st}_2, \text{GetInst} \]

\[ \text{op}, i, \text{data}, \text{st}_2 \]

\[ 16 (\text{op} = \text{write}) \]

\[ \hat{D}[i] = \text{data} \]

\[ \]

\[ \text{val/no} \]

\[ \text{Done?} \]

\[ \text{yes/no} \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]

\[ \]
Generic Protocol:
- protocol that can securely evaluate any function in some specified class of functions
  \( \rightarrow \) e.g. finite automaton

Special-purpose / tailored protocol:
- designed specifically for some function \( f \)

Problem:
Set intersection:
each \( P_i \) has a set \( S_i \subseteq \{0,1\}^n \)
Parties want to compute \( S_1 \cap S_2 \)

ENP '04:
- homomorphic encryption-based solution

Observation:
Let \( P(x) = \sum_i a_i x^i \)
Given \( [a_1], [a_2], \ldots, [a_e], y \)
Can compute \( [P(y)] \)
observation:

$$P_1\left(\{x_1, \ldots, x_n\}\right) \quad P_2\left(\{x_1, \ldots, x_n\}\right)$$

Encode its set as 0's on a polynomial $Q$.

i.e. $Q(x) = \prod_i (x - x_i)$

$$Q(x) = \sum_i a_i \cdot x^i$$

$$pk, [a_0], [a_1], \ldots, [a_n]$$

compute $[Q(x_1)], \ldots, [Q(x_n)]$

$$compute \left[ r, Q(x_1) + x_1 \right]$$

$$\left[ r, Q(x_n) + x_n \right]$$

$$\left[ r_0 a_0 + x_1, \ldots, r_0 a_0 + x_n \right]$$

decrypt...