Paljastavad küsimused ehk ligikaudne otsing

Sven Laur
swen@math.ut.ee

Tartu Ülikool

Road-map

- What is approximate matching?
- What can we do with approximate matching?
- There is no easy solution to secure approximate matching.
- Secure randomised affine transformation.
- More efficient settings
- There is no easy solution.
- Conclusion

Motivation

How to sell the same thing twice? — Start a service!

- But collecting and systematising the information is expensive.

- And some services violate privacy.

Usually privacy is not a main concern, except

- if queries reveal delicate information;

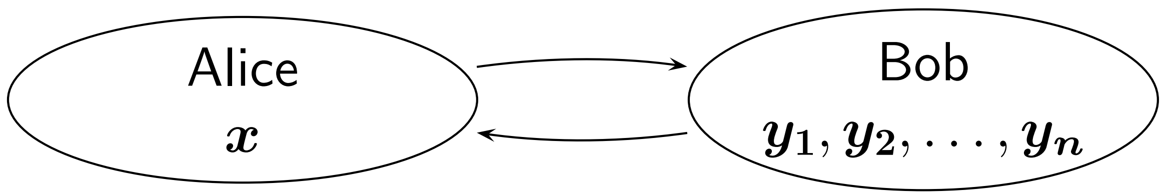
- if a leakage causes explicit economical expenses.

Onion-routing does not guarantee privacy, since

- a query itself can reveal the personality;

- a query itself can contain useful information.

Approximate matching



$$d_0 = \min_i d(\boldsymbol{x}, \boldsymbol{y_i})$$
$$i_0 = \operatorname*{argmin}_i d(\boldsymbol{x}, \boldsymbol{y_i})$$

Security considerations

- Bob should learn nothing about query vector $\boldsymbol{x}$.

- Alice should learn nothing about the database, except the match distance d_0 and the match number i_0.

Efficiency considerations

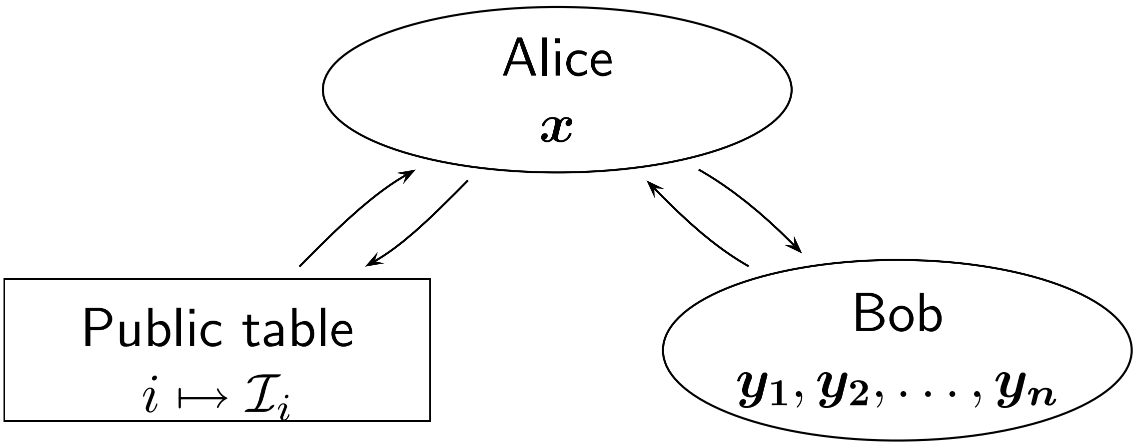
- Communication complexity should be $\mathbf{poly}(\log n)$.

- Computational complexity should be $\mathbf{poly}(n)$.

Example application: Symptom-action type databases

Bob has accumulated knowledge in the following form.

- A symptom $\boldsymbol{y}_i$ and an appropriate action $\mathcal{I}_i$.
- Nearest neighbourhood search gives appropriate action for an unknown $\boldsymbol{x}$.



- The protocol does not require private information retrieval!
- An approximate matching might be more efficient!

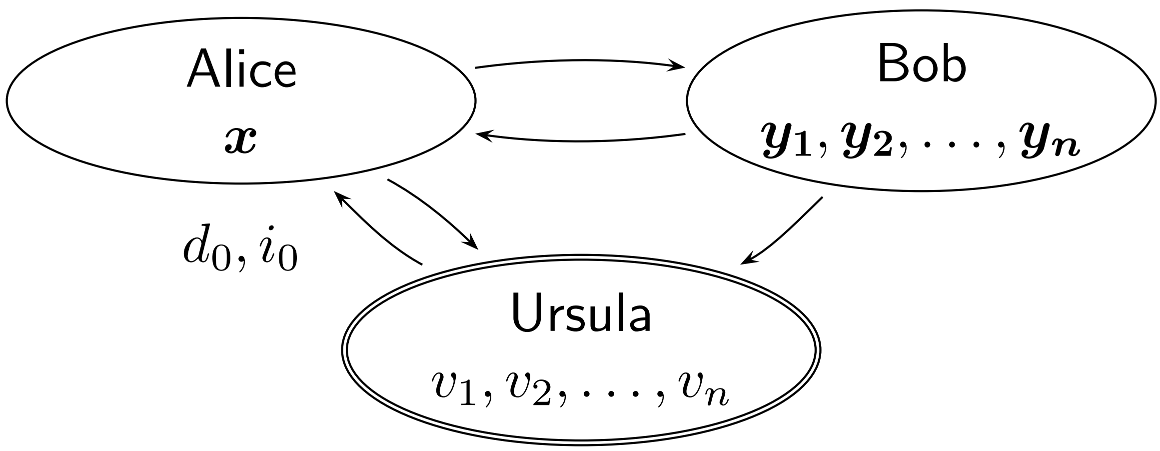
Further analysis

- Approximate matching requires a secure evaluation of minimum

$$d_0 = \min_i d(\boldsymbol{x}, \boldsymbol{y_i}).$$

- Currently, no efficient and cryptographically secure minimum finding protocols are known.

- To bypass the problem, we include a trusted third party Ursula.



- Naive implementations yield communication complexity $\Theta(n)$.

From distance to scalar product

For the Euclidean distance, we must calculate

$$d_i = (\boldsymbol{x} - \boldsymbol{y_i})^2 = \boldsymbol{x}^2 - 2\boldsymbol{x} \cdot \boldsymbol{y_i} + \boldsymbol{y_i}^2.$$

Transformation

$$\begin{aligned} \boldsymbol{x} &= (x_1, \ldots, x_m) \mapsto \boldsymbol{x'} = (-2x_1, \ldots, -2x_n, 1), \\ \boldsymbol{y} &= (y_1, \ldots, y_m) \mapsto \boldsymbol{y'} = (y_1, \ldots, y_n, y_1^2 + \cdots + y_m^2) \end{aligned}$$

gives

$$\boldsymbol{x'} \cdot \boldsymbol{y'_i} = -2\boldsymbol{x} \cdot \boldsymbol{y_i} + \boldsymbol{y_i}^2$$

and thus

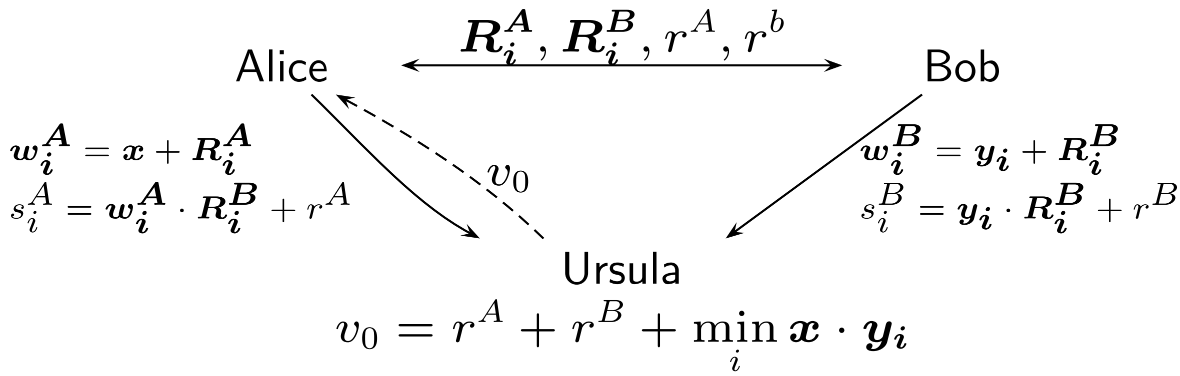
$$\begin{aligned} d_0 &= \boldsymbol{x}^2 + \min_i \boldsymbol{x'} \cdot \boldsymbol{y'_i}, \\ i_0 &= \operatorname*{argmin}_i \boldsymbol{x'} \cdot \boldsymbol{y'_i}. \end{aligned}$$

MinDASP protocol (Du and Atallah)

The protocol hinges on the fact that

$$
\begin{aligned}
&(\boldsymbol{x} + \boldsymbol{R_i^A}) \cdot (\boldsymbol{y} + \boldsymbol{R_i^B}) \\
&\qquad = \boldsymbol{x} \cdot \boldsymbol{y_i} + \underbrace{\boldsymbol{R_i^A} \cdot \boldsymbol{y_i}}_{s_i^B - r^B} + \underbrace{(\boldsymbol{x} + \boldsymbol{R_i^A}) \cdot \boldsymbol{R_i^B}}_{s_i^A - r^A}
\end{aligned}
$$

Therefore, we get a straightforward protocol



- Ursula gets additive shares $v_i = \boldsymbol{x} \cdot \boldsymbol{y_i} - r^A - r^B$.
- For one database element all values are perfectly masked, but this is not true for many items.

The attack

Note that

$$s_i^B = \boldsymbol{R_i^A} \cdot \boldsymbol{w_i^A} + r^B = (\boldsymbol{w_i^A} - \boldsymbol{x}) \cdot \boldsymbol{w_i^B} + r^B$$

and therefore

$$(\boldsymbol{w_i^B} - \boldsymbol{w_1^B}) \cdot \boldsymbol{x} = \boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} - \boldsymbol{w_1^A} \cdot \boldsymbol{w_1^B} - (s_i^B - s_i^A).$$

Since Ursula knows everything except $\boldsymbol{x}$, she can compose a system of linear equations $M\boldsymbol{x} = \boldsymbol{z}$.

- We calculated the exact probability that $M\boldsymbol{x} = \boldsymbol{z}$ has a unique solution. The probability is too big.

- We offered two bug-fixes: a slight modification of the protocol and a secure scalar product protocol via homomorphic encryption.

- The latter reduces communication complexity more than four times, however the computational complexity rises.

General result

It is unreasonable to assume that Ursula knows nothing about the database.

- Some database elements might be publicly known.
- During the longterm use of database, some vectors might leak.

The protocol should remain secure if less than $\tau = \Theta(n)$ database elements are known to Ursula.

Theorem 1. *All protocols, where Ursula obtains additive shares $v_i = \boldsymbol{x} \cdot \boldsymbol{y_i} + r$, are insecure regardless of used sub-protocols.*

- If Ursula knows database vectors $\boldsymbol{y_1}, \ldots, \boldsymbol{y_k}$, then she can construct a system of linear equations.
- Under the random database assumption the system has unique solutions with high enough probability.
- Otherwise the security depends on the specific database!

The MinAffineSP protocol: The working principle

- Public parameters are integers p, q, t_{max}, t_{tmin}, so that $t_{min} \approx 2^{160}$ and $q(t_{max}+1) < p$.

- Alice and Bob jointly fix a random multiplier $t \in [t_{min}, t_{max}]$.

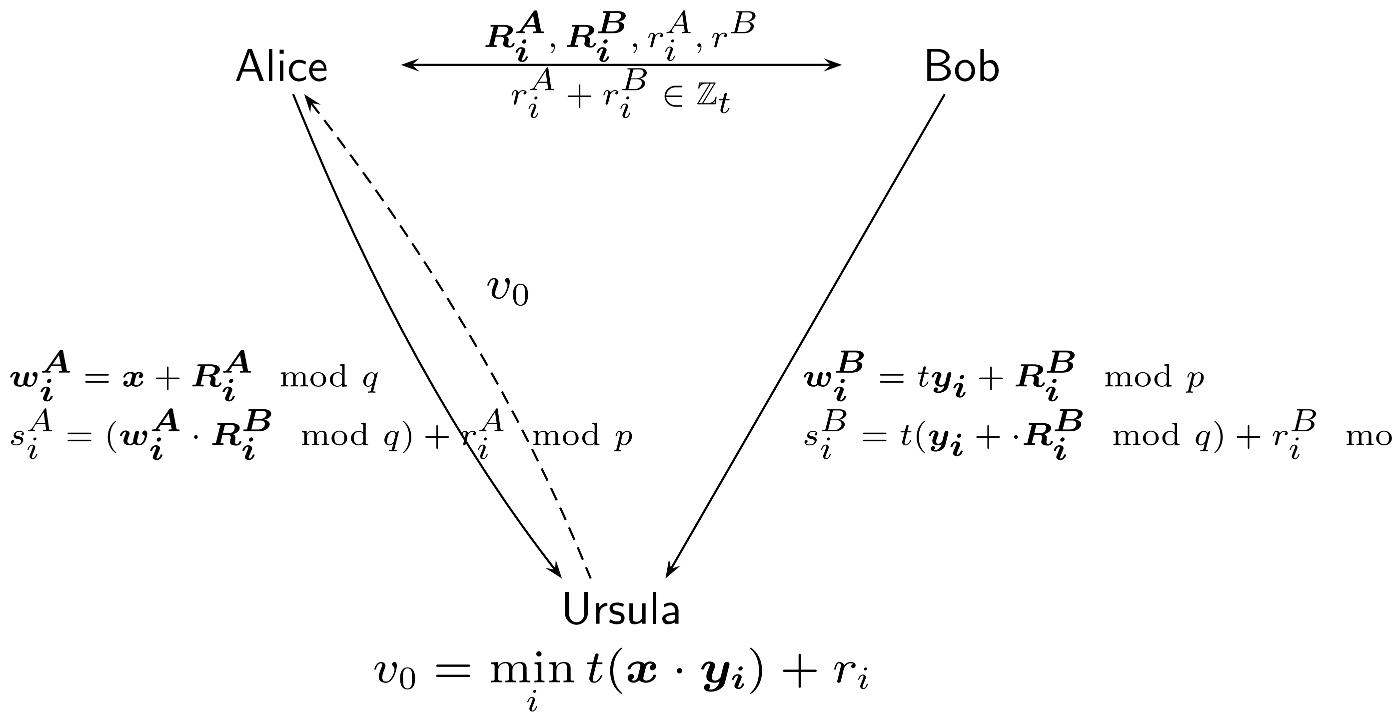
- Ursula obtains shares $v_i = t(\boldsymbol{x} \cdot \boldsymbol{y_i} \mod q) + r_i$, where $r_i \in \mathbb{Z}_t$, that are randomly permuted.

- The smallest share v_0 still corresponds to the minimal scalar product.

- Alice can eliminate the randomness

$$\min_i \boldsymbol{x} \cdot \boldsymbol{y_i} = \left\lfloor \frac{v_0}{t} \right\rfloor .$$

The MinAffineSP protocol: The implementation



After some tedious calculations

$$
\begin{aligned}
v_i &= (\boldsymbol{w_i^A} \cdot \boldsymbol{w_i^B} \mod q) - s_i^A - s_i^B \\
&= t(\boldsymbol{x} \cdot \boldsymbol{y_i} \mod q) + r_i \mod p.
\end{aligned}
$$

The MinAffineSP protocol: The preliminary security analysis

To break the protocol, Ursula must solve the system of equations

$$tx_i + r_i = z_i, \qquad t \in [t_{min}, t_{max}],$$

where $x_i \in \mathbb{Z}_q, r_i \in \mathbb{Z}_t$ are unknown.

- The values r_i are uniformly distributed for correct t and not generally uniformly distributed for $t' \nmid t$.

- Small differences of $v_i - v_j = \Delta x_{i,j} t + r_i - r_j$ can suggest the values for t. If $\Delta x_{i,j} = 1$ then we have a slight probability peak at t.

- Linear combinations $a_1 v_1 + \cdots + a_m v_m$ with small coefficients of a_i are more restrictive, since the true random term $a_1 r_1 + \cdots + a_m r_m$ converges to the normal distribution.

- But the probabilistic reduction increases the uncertainty by $m^{1/2}$ times.

More efficient settings: Secure storage outsourcing problem

Consider a scenario

- First Alice outsources the database to Bob.

- Afterwards Alice needs match distances

$$\min_i (\boldsymbol{x_j} - \boldsymbol{y_i})^2 .$$

Security considerations

- Bob should learn nothing about database vectors $\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}$ and query vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}$.

- The protocol should remain secure even if Bob knows $\tau = \Theta(n)$ vectors $\boldsymbol{x_j}$ or $\boldsymbol{y_i}$.

Efficiency considerations

- The protocol should have only $\mathcal{O}(1)$ rounds.

General result

Proposed solutions use additive sharing, that is Bob can calculate $s_{ij} = (\boldsymbol{x_j} - \boldsymbol{y_i})^2 + r_j$.

- The second security goal—a resistance against leaking vectors—is not satisfied.

- The difference matrix $\Delta S = (\Delta s_{ij})$, where

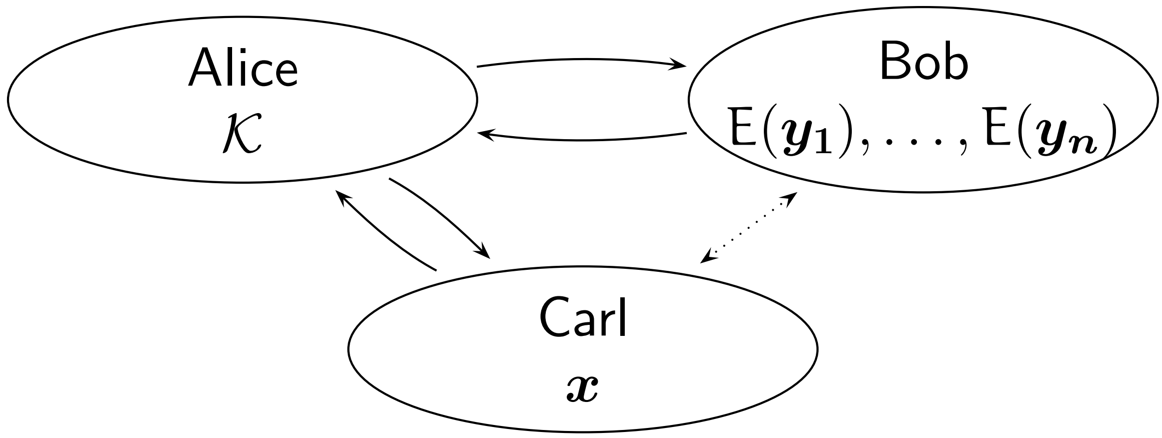
  $$\Delta s_{ij} = s_{ij} - s_{i1} - s_{j1} + s_{11} = -2\boldsymbol{\Delta x_j}\boldsymbol{\Delta y_i},$$

  reveals linear information.

- If the database does not belong to the hyper-plane then the matrix columns reveal linear dependencies between query differences $\boldsymbol{\Delta x_j} = \boldsymbol{x_j} - \boldsymbol{x_0}$.

- The matrix rows reveal linear dependencies between the differences $\boldsymbol{\Delta y_i} = \boldsymbol{y_i} - \boldsymbol{y_1}$.

More efficient settings: Secure storage and computing outsourcing problem



- First Alice outsources the database to Bob.
- Afterwards Carl composes queries with the help of Alice.
- Alice helps Carl to decode the answers.

Practical requirements

Security considerations

- Bob should learn nothing about the database vectors $\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}$ and query vectors $\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}$.
- Alice should learn nothing about the query vectors.
- Collusion between Bob and Carl is allowed
- The protocol should remain secure even if Bob knows $\tau = \Theta(n)$ vectors $\boldsymbol{y_i}$.

Efficiency considerations

- The protocol should have only $\mathcal{O}(1)$ rounds.

General result

Theorem 2. *The protocols, where Bob obtains linear shares* $s_{ij} = \alpha_j(\boldsymbol{x_j} - \boldsymbol{y_i})^2 + r_j$, *are insecure regardless of used sub-protocols.*

- The second security goal—resistance against leaking vectors—is not satisfied.
- Carl and Bob can restore originals of database elements by gradient search.
- More subtle attacks are possible.

Conclusions: Curse of linearity

- It is hard to find the minimum when shares do not have the linear form $\alpha x + r$.
- But the linearity opens door to relatively simple attacks based on linear algebra.
- The linear transformation is not safe a way to hide data.