Privacy-Preserving Search of Similar Patients in Genomic Data

Gilad Asharov

CORNELL TECH

Shai Halevi

IBM

Yehuda Lindell

BIU Center for Research in Applied Cryptography and Cyber Security

Tal Rabin

IBM

Secure Computation

- Computation on private inputs without revealing anything but the output

- **Applications**:
  - Run machine learning algorithms on distributed databases
  - Blockchains
  - Protecting credentials, cryptographic keys
  - Protecting biometrics
  - Genomics
  - Social networks

Secure Computation

Generic Protocols

Protocols for specific tasks

- **This talk:**
  - Design of a secure protocol for a specific task in genomics
  - Demonstrating several design principles
    - Pushing most of the computation to the preprocessing

hours → **seconds**

The Task

- A doctor has the **genome sequence** of her patient
  - Want to use it to help diagnosis/treatment options
- Compare sequence against a **database** with many sequences
  - Each sequence with a list of conditions
- **Want to identify the few DB sequences closest to the patient's**
  - Get the list of associated conditions

Challenge:
Doing this while protecting privacy
(of the patient as well as the patients in the DB)

A Motivating Scenario: Cancer Patients

- Comparing genome with the one in patient's tumour will help **pinpoint** which mutations are behind the disease

2017	50,000
***2030**	248,000,000

Global Alliance for Genomics & Health
Collaborate. Innovate. Accelerate.

iDASH

IDASH PRIVACY & SECURITY WORKSHOP 2016

Track 2: Privacy-Preserving Search of Similar Cancer Patients across Organizations (secure multiparty computing)

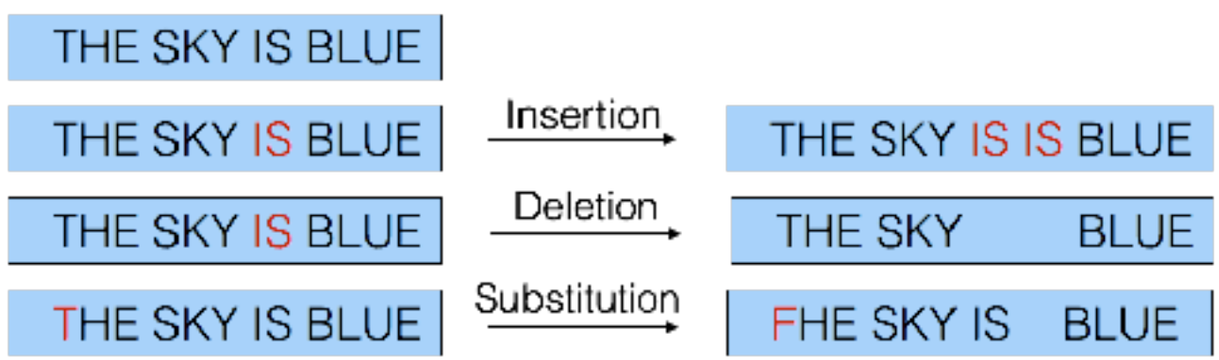
The scenario of this track is to find top-k most similar patients in a database on a panel of genes. The similarity is measured by the **edit distance** between a **query** sequence and **sequences in the database**. We expect participating teams come up with different algorithms that can provide good approximation to the actual edit distance and also be efficient. (data link)

Edit Distance

- Counting the minimum number of basic operations required to transform one string into the other

```
T T T C T T T A A T G G T T A T
```

```
T T T C T T   A A T A G T T A G A A
```



- $O(n^2)$ comparisons
- $O(nd)$ if we have a-priory bound d on the distance

The Challenge Database

- 500 sequences, each of size ~3500
- Taken from a **high-diversity** region (gene ZNF717, Chromosome 3)
- Distance between individuals ~ 5%
- Each ED requires at least 3500x200~**700,000** comparisons
  - Even if we have a-priory bound ED < 200
  - These are~ **50M** gates
- For computing 500 EDs = **25B** gates
- Would take several **hours**
  - Even when using current state-of-the-art secure computation

Our Work

- “Domain specific” edit distance approximation
- Secure-computation protocol for computing it (semi-honest)
- **Very accurate**
  - Tested on several different regions with high-diversity
  - Returns the exact set on >98% times,
    Very good approximation on the remaining <2%
- **Very fast**
  - Most of the work is done during preprocessing, on “cleartext”
  - <1.5 seconds per query, after ~11sec of preprocessing

- **Won the iDash competition** (8 submitted solutions)

Related W

- Works by reducing edit distance to set interaction
- Only useful in “low diversity” regions

- **Similar Patient Query:**
  - Wang, Huang, Zhao, Tang, Wang, Bu
    Efficient genome-wide, privacy-pr erving of similar patients query based on private edit distance

- **Surveys:**
  - Naveed, Aydaym Clayton, Fellay, Gunter, Hubaux, Malin, Wang
    Survey: Privacy in the genomic era
  - Akgu¨n, Bayrak, Ozer, and Sag˘ırog˘lu
    Privacy preserving processing of genomic data

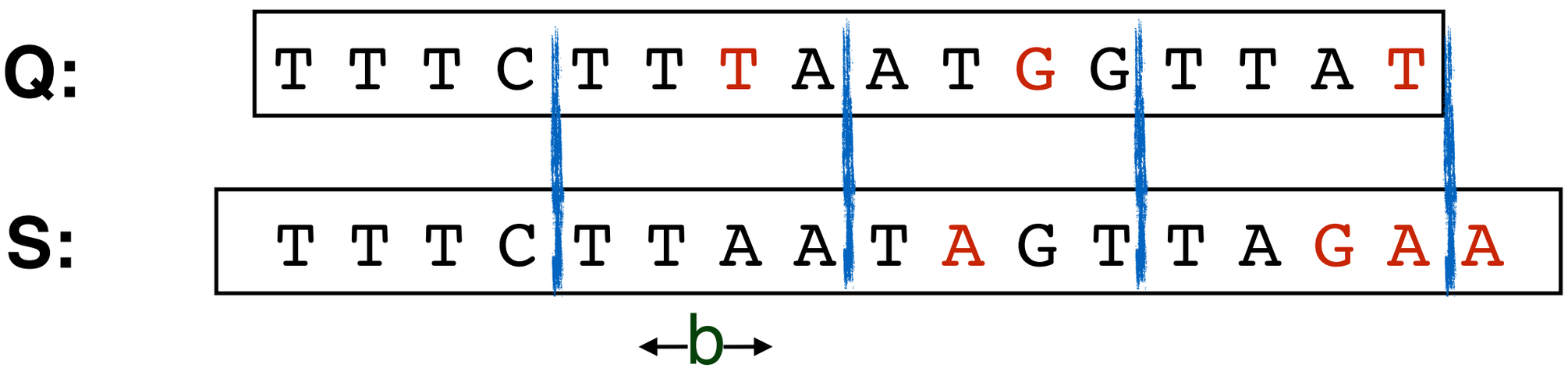
- **Security implication of computing approximations:**
  Feigenbaum, Ishai, Malkin, Nissim, Straus, Wright

- **Concurrent works:**
  - Al Aziz, Alhadid, Mohammed
    Secure approximation of edit dis
  - Zhu, Huang
    Efficient privacy preserving gene dit-distance and beyond

- Competitors in the iDash competition

Our Protocol

Efficient “Approximation”

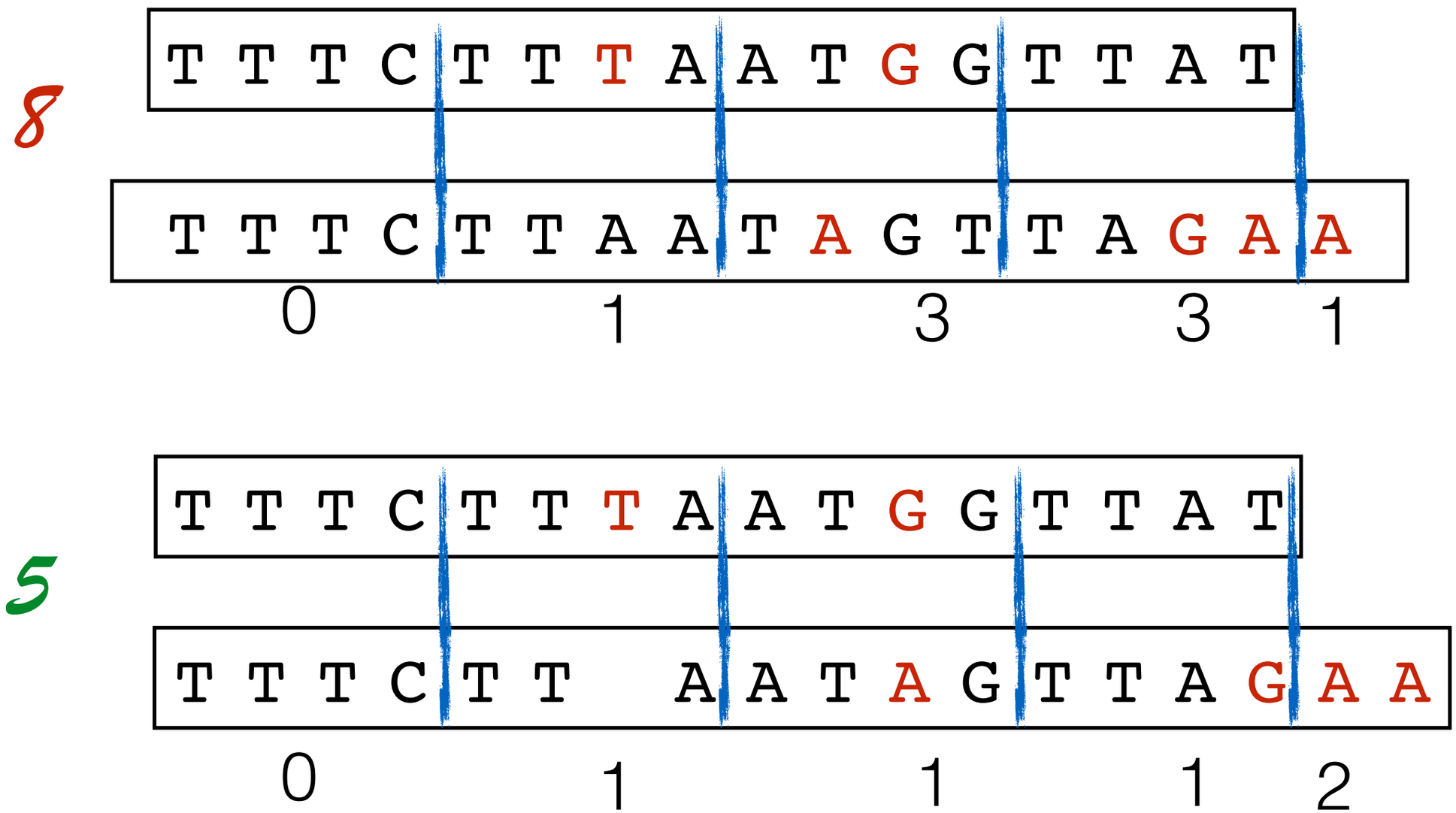


$$ApproxED(Q,S)=\sum_i ED(Q_i,S_i)$$

$$n/b * O(b^2) = O(nb)$$

Becomes linear!

Efficient, but Not Good



Clearly, the break points are important

How do we know where to split the sequence?

We Align According to the Reference Genome!

- We utilize a **reference genome**
  - Publicly available online
  - Was assembled from several donors
  - Aim: to use a single, preferred tiling path to produce a single consensus representation of the genome
- We run a full edit-distance between the **sequence** and the **reference genome**
- Break the **reference genome** to fix-width blocks
- Break the **sequence** to variable-width blocks that align with the **reference sequence** blocks

Seq A C A C A C T A

Ref: A G C A C A

Seq: A C A C A C T A

Ref: A G C A C A

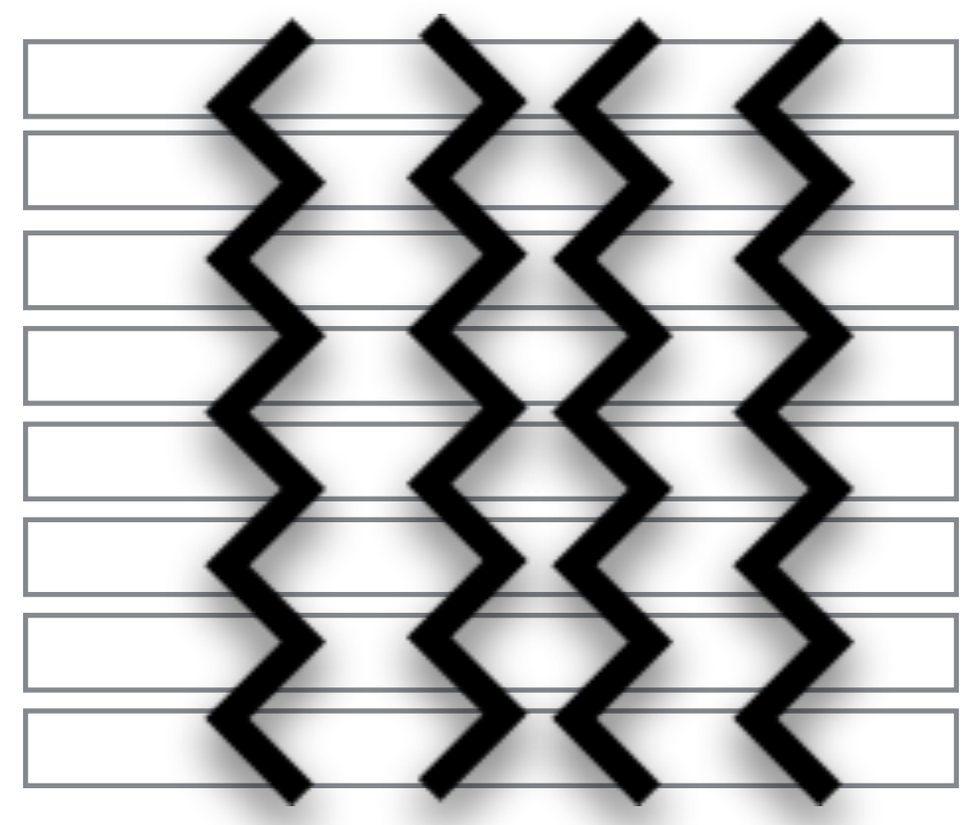
The Genomic Distribution

Client
a single query

1 query

DB
many DNA sequences

500 sequences
|seq| ~ 3500



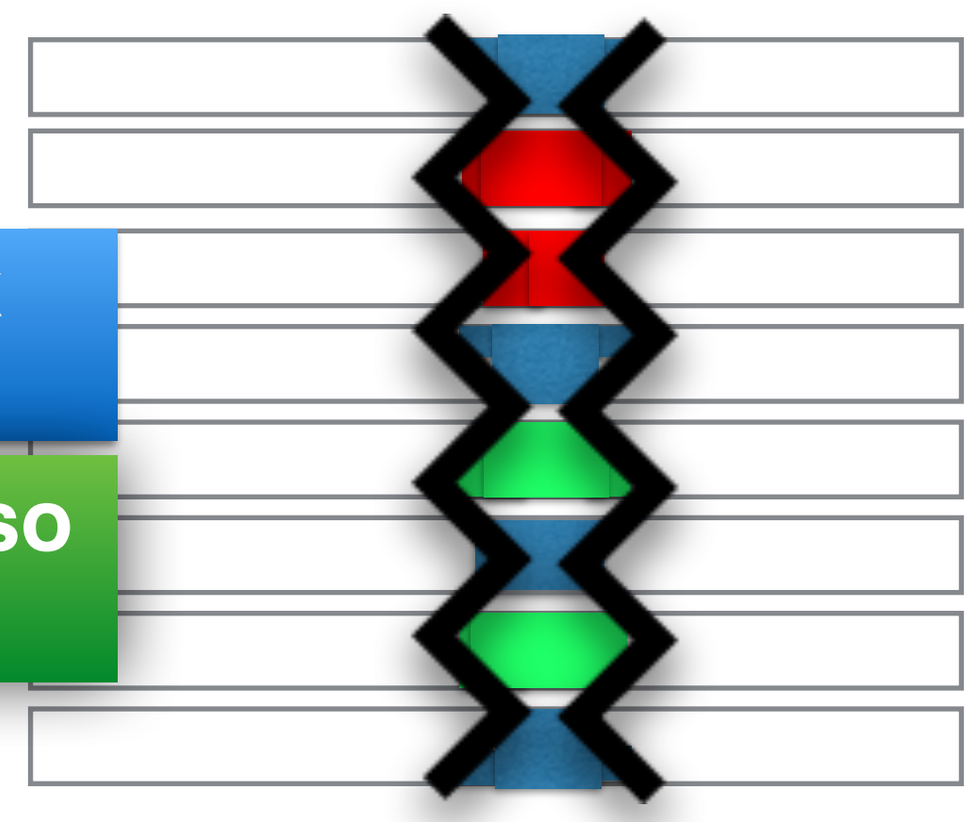
The Genomic Distribution

Client
a single query

1 query

DB
many DNA sequences

500 sequences
|seq| ~ 3500



Very few distinct values in each block across all the DB (500 —> ~10)

In most cases the query block is also one of these values!

The Genomic Distribution

Client
a single query

1 query

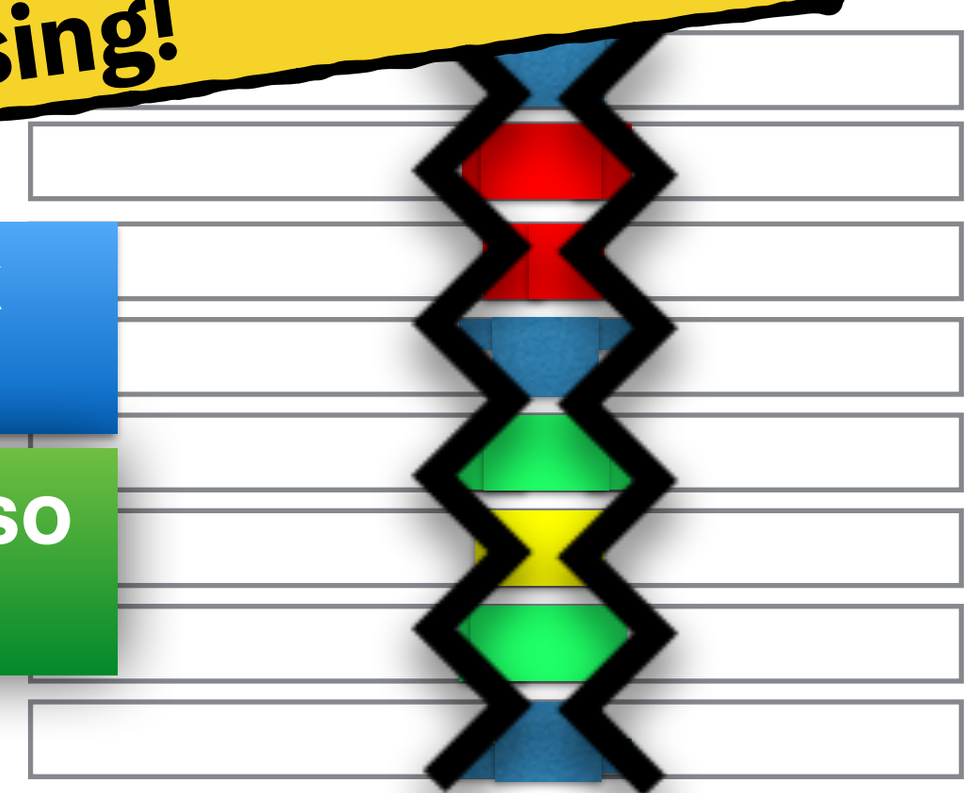
DB
many DNA sequences

500 es

We can push almost all computation to the preprocessing!

Very few distinct values in each block across all the DB (500 —> ~10)

In most cases the query block is also one of these values!



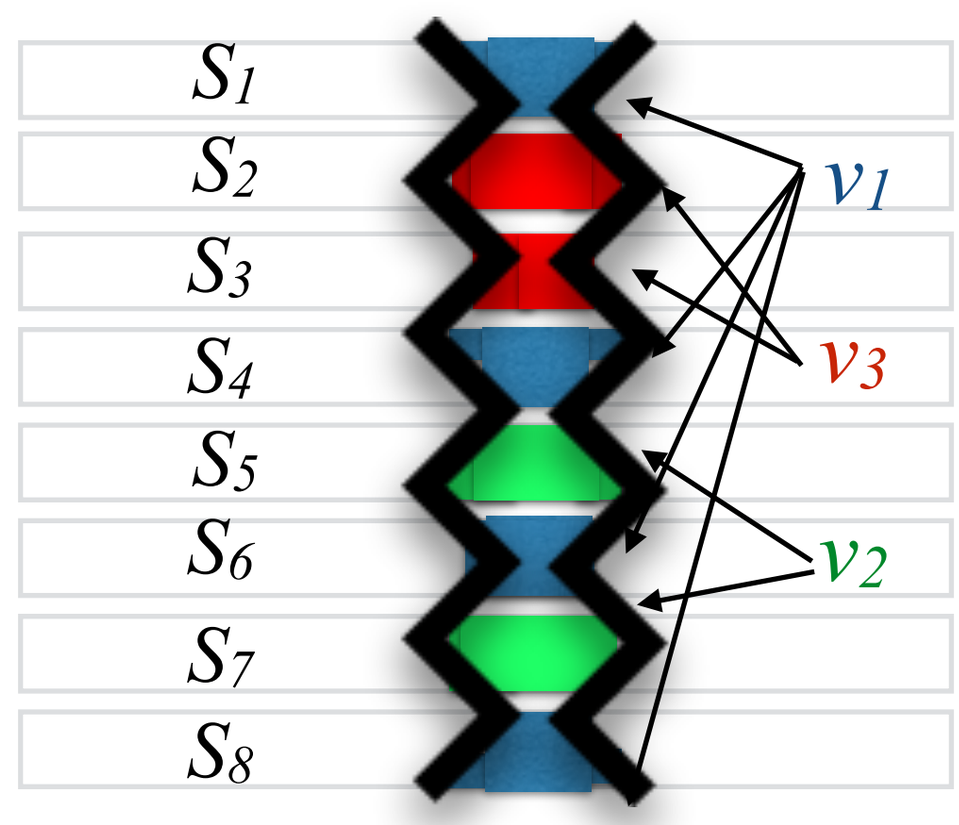
Server Preprocessing

Block I:

	$\{v_1,$	$v_2,$	$v_3\}$
S_1	0	2	1
S_2	1	3	0
S_3	1	3	0
S_4	0	2	1
S_5	2	0	3
S_6	1	2	1
S_7	2	0	3
	…	…	…
	$\Delta_{1,1}$	$\Delta_{1,2}$	$\Delta_{1,3}$

notation

$\Delta_{i,u}$:
a vector of length |DB|
The contribution of the **i'th** block to the approximation if the **i'th** block of the query is the **u'th** value



Server Preprocessing

Block I:

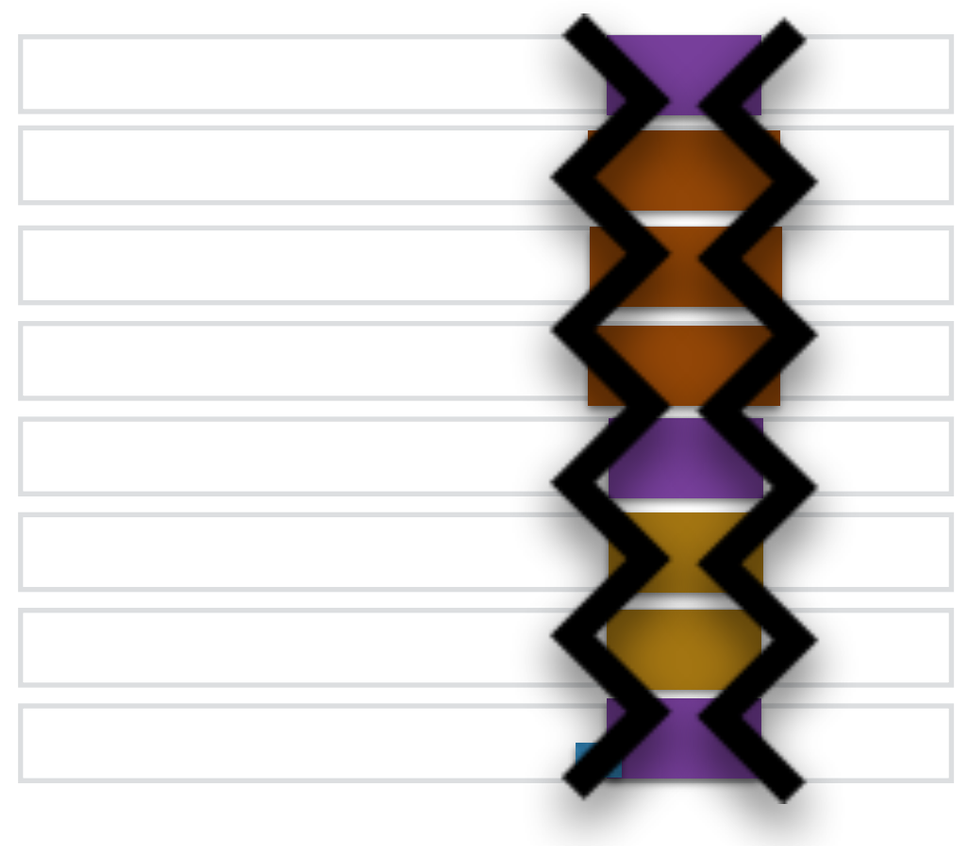
$\{v_1, v_2, v_3\}$

0	2	1
1	3	0
1	3	0
0	2	1
2	0	3
1	2	1
2	0	3
…	…	…
$\Delta_{1,1}$	$\Delta_{1,2}$	$\Delta_{1,3}$

Block II:

$\{u_1, u_2, u_3\}$

0	1	1
1	0	1
1	0	1
1	0	1
0	1	1
1	1	0
1	1	0
…	…	…
$\Delta_{2,1}$	$\Delta_{2,2}$	$\Delta_{2,3}$



Online Computation

Block I:

$\{v_1,$	$v_2,$	$v_3\}$
0	2	1
1	3	0
1	3	0
0	2	1
2	0	3
1	2	1
2	0	3
…	…	…
$\Delta_{1,1}$	$\Delta_{1,2}$	$\Delta_{1,3}$

Block II:

$\{u_1,$	$u_2,$	$u_3\}$
0	1	1
1	0	1
1	0	1
1	0	1
0	1	1
1	1	0
1	1	0
…	…	…
$\Delta_{2,1}$	$\Delta_{2,2}$	$\Delta_{2,3}$

The query:

…

1) **Break it into blocks (ref genome)**
2) Compare each block to the corresponding set of values in the DB

Online Computation

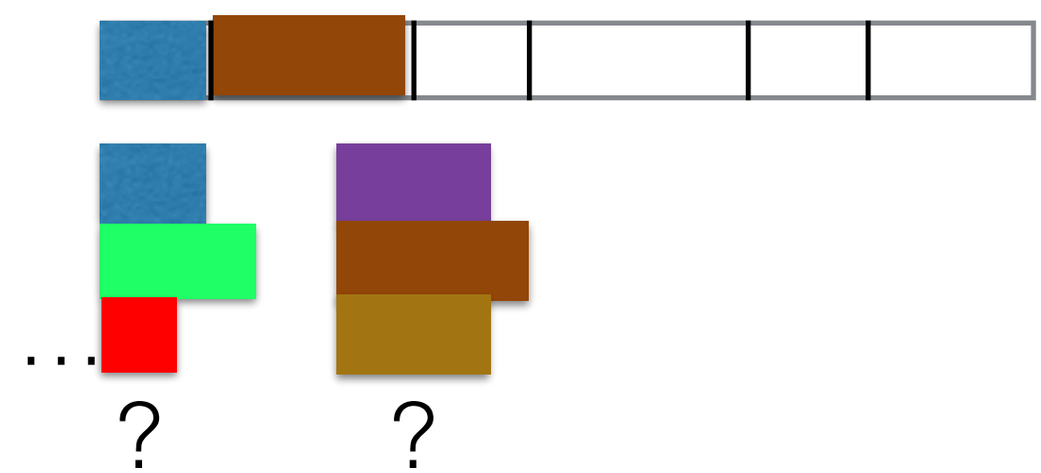
Block I:

$\{v_1,$	$v_2,$	$v_3\}$
0	2	1
1	3	0
1	3	0
0	2	1
2	0	3
1	2	1
2	0	3
…	…	…
$\Delta_{1,1}$	$\Delta_{1,2}$	$\Delta_{1,3}$

Block II:

$\{u_1,$	$u_2,$	$u_3\}$
0	1	1
1	0	1
1	0	1
1	0	1
0	1	1
1	1	0
1	1	0
…	…	…
$\Delta_{2,1}$	$\Delta_{2,2}$	$\Delta_{2,3}$

The query:



1) Break it into blocks (ref genome)
2) **Compare each block to the corresponding set of values in the DB**

Online Computation

Block I:

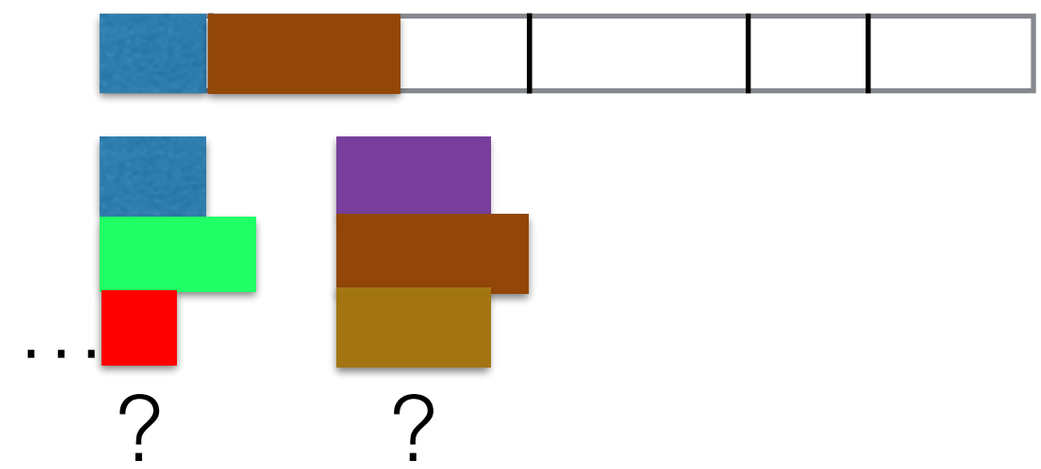
$\{v_1,$	$v_2,$	$v_3\}$
0	2	1
1	3	0
1	3	0
0	2	1
2	0	3
1	2	1
2	0	3
…	…	…

Block II:

$\{u_1,$	$u_2,$	$u_3\}$
0	1	1
1	0	1
1	0	1
1	0	1
0	1	1
1	1	0
1	1	0
…	…	…

vec	$\Delta_{1,1}$	$\Delta_{1,2}$	$\Delta_{1,3}$	$\Delta_{2,1}$	$\Delta_{2,2}$	$\Delta_{2,3}$
bits	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$

The query:



notation

$x_{i,u}$: a bit

The **i'th** block of the query = the u'th value?

ApprxED(Q,DB)=

$$\sum_i \sum_u x_{i,u} \Delta_{i,u}$$

The Secure Protocol

Block I:

$\{v_1, v_2, v_3\}$

0	2	1
1	3	0
1	3	0
0	2	1
2	0	3
1	2	1
2	0	3
…	…	…

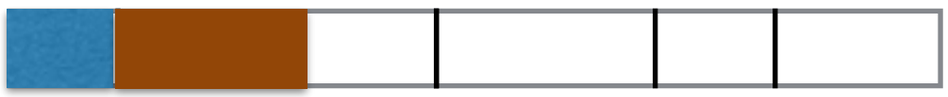
Block II:

$\{u_1, u_2, u_3\}$

0	1	1
1	0	1
1	0	1
1	0	1
0	1	1
1	1	0
1	1	0
…	…	…

vec	$\Delta_{1,1}$	$\Delta_{1,2}$	$\Delta_{1,3}$	$\Delta_{2,1}$	$\Delta_{2,2}$	$\Delta_{2,3}$
bits	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$

The query:



1) Break the query to blocks
2) Using **Yao's garbled circuit**: Compute the (shares of) bits $x_{i,u}$
3) Using **oblivious transfer**, obtain shares of $x_{i,u}\Delta_{i,u}$
4) Using local computation, obtain shares of $\mathrm{ApprxED}(Q,DB)=\sum_i \sum_u x_{i,u}\Delta_{i,u}$
5) k-min using a naive circuit (using **Yao's garbled circuit**)

Accuracy and Performance

- Tested on various databases, different sizes, different genes
  - Tested also on fake synthesized data for scaleability
- **Accuracy**
  - >98% successfully returns the exact k-set
  - <2% returns someone that is at most 1 away from the true result
- **Bandwidth**: < 80MB

Gene	Samples	Length	Preprocessing (sec)	Online (sec)	#AND Gates
ZNF717	500	3470	11.86	1.22	1,506,625
CDC27P2	100	1950	0.91	0.45	650,018
TEKT4P2	50	2087	0.69	0.45	648,308

25,000,000,000 **AND gates**

↓

1,500,000 **AND gates**

Conclusions

- We “reduced” edit distance to simple comparisons
- We demonstrate that MPC can achieve such high performance in specific (important) problem
  - But such “tricks” are possible also in other problems?
  - Encourage to consider using MPC in places where initially it looks too expensive
- **Acknowledgments**
  - Shalev Keren, Meital Levy, Assi Barak

Thank you!