Protecting Patient Privacy in Genomic Analysis

David Wu
Stanford University

based on joint works with:
Gill Bejerano, Bonnie Berger, Johannes A. Birgmeier, Dan Boneh, Hyunghoon Cho, and Karthik A. Jagadeesh
Rare Disease Diagnosis

What gene causes a specific (rare) disease?

Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]
## Rare Disease Diagnosis

Each patient has a list of 200-400 rare variants over ≈20,000 genes.

<table>
<thead>
<tr>
<th>Gene</th>
<th>A1BG</th>
<th>ZZZ3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Each patient has a vector $\mathbf{v}$ where $v_i = 1$ if patient has a rare variant in gene $i$.

**Goal:** Identify gene with most variants across all patients.

- Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes.

*Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]*
Rare Disease Diagnosis

<table>
<thead>
<tr>
<th>Gene</th>
<th>A1BG</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ZZZ3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Each patient has a vector $\mathbf{v}$ where $v_i = 1$ if patient has a rare variant in gene $i$.

Goal: Identify gene with most variants across all patients.

Works well for Mendelian (monogenic) diseases (estimated to affect $\approx$10% of individuals).

Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over $\approx$20,000 genes.
Patients often in geographically-diverse locations

Question: Can we perform this computation **without** seeing complete patient genomes?
Rare Disease Diagnosis

Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes

Patients “secret share” their data with two non-colluding hospitals

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]
Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes

Hospitals run a multiparty computation (MPC) protocol on pooled inputs

Patients “secret share” their data with two non-colluding hospitals

Rare Disease Diagnosis

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]
Rare Disease Diagnosis

Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes

MPC Protocol

Top variants (sorted):

$\text{KMT2D, COL6A1, FLNB}$

Known cause of disease

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]
Rare Disease Diagnosis

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]

Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes

Top variants (sorted): KMT2D, COL6A1, FLNB

Other variants that the patients possess are kept hidden.
General techniques apply to many different scenarios for diagnosing Mendelian diseases.

- Identify causal gene for a rare disease given a small patient cohort.
- Patients with Kabuki Syndrome.
- Identify patients with the same rare functional mutation at two different hospitals.
- Identify rare functional variants that are present in the child but in neither of the parents.
Rare Disease Diagnosis

Jagadeesh-W-Birgmeier-Boneh-Bejerano [Science 2017]

General techniques apply to many different scenarios for diagnosing Mendelian diseases

- Identify causal gene for a rare disease given a small patient cohort
- Patients with Kabuki Syndrome
- Identify rare functional variants that are present in the child but in neither of the parents

Simple frequency-based algorithms, but techniques enabled us to discover a previously unidentified pathogenic variant
Experimental benchmarks for identifying causal gene in small disease cohort
  • Simulated two non-colluding entities with 1 server on East Coast and 1 on West Coast
Experimental benchmarks for identifying causal gene in small disease cohort

- Simulated two non-colluding entities with 1 server on East Coast and 1 on West Coast

For many rare disease diagnosis scenarios, disease cohort size can be very small (e.g., 5-10 patients)
Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes.
Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes. Techniques apply to general computations over private data.
Yao’s Protocol for Two-Party Computation
Yao’s Protocol for Two-Party Computation \cite{Yao82}

Classic protocol for two-party computation

Security guarantee: everything the parties learn can be inferred from the output and their individual inputs
Yao’s Protocol for Two-Party Computation [Yao82]

Step 1: Model computation as a Boolean circuit

Private inputs

Party 1 “garbler”

Party 2 “evaluator”

Party 1

Party 2

Output

AND

NAND

AND
Yao’s Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Garbler chooses two different encryption keys for every wire in the circuit.

Each key is associated with a possible wire value.
Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Idea: Encrypt the output key (for the output wire) with the two input keys (for the input wires)

Garbler constructs a garbled truth table for each gate in the circuit
Yao’s Protocol for Two-Party Computation [Yao82]

Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

\[ \text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right) \]

Garbler constructs a garbled truth table for each gate in the circuit.
Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Garbler constructs a garbled truth table for each gate in the circuit
Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Garbler constructs a garbled truth table for each gate in the circuit.

\[
\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right) \\
\text{Enc} \left( k_0^{(1)}, \text{Enc} \left( k_1^{(2)}, k_0^{(\text{out})} \right) \right) \\
\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_0^{(2)}, k_0^{(\text{out})} \right) \right) \\
\text{Enc} \left( k_1^{(1)}, \text{Enc} \left( k_1^{(2)}, k_1^{(\text{out})} \right) \right)
\]
Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Garbler constructs a garbled truth table for each gate in the circuit

Garbled truth table randomly permuted
**Yao’s Protocol for Two-Party Computation** [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Garbler constructs a garbled truth table for each gate in the circuit

![Garbled truth table randomly permuted]

**Invariant:** Given just a single key for each input wire, evaluator can learn a single key for the output wire

Garbler constructs a garbled truth table for each gate in the circuit
Yao’s Protocol for Two-Party Computation [Yao82]

Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

\[ \text{Garbled truth table randomly permuted} \]

\[ \text{Invariant: Given just a single key for each input wire, evaluator can learn a single key for the output wire} \]

\[ k_1^{(1)}, k_0^{(2), (\text{out})} \]
Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

\[ \text{Enc}\left(k_1^{(1)}, \text{Enc}\left(k_0^{(2)}, k_0^{(\text{out})}\right)\right) \]

Garbled truth table randomly permuted

Invariant: Given just a single key for each input wire, evaluator can learn a single key for the output wire

\( k_0^{(\text{out})} \) is just a symmetric key – does not reveal what the output bit is
Step 2: Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Invariant: Given just a single key for each input wire and a garbled table, evaluator can learn a single key for the output wire.
Yao’s Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Invariant: Given just a single key for each input wire and a garbled table, evaluator can learn a single key for the output wire.
Yao’s Protocol for Two-Party Computation \([\text{Yao82}]\)

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

**Invariant:** Given just a single key for each input wire and a garbled table, evaluator can learn a single key for the output wire

Include decoding table to map output keys to output values.
Yao’s Protocol for Two-Party Computation [Yao82]

**Step 2:** Garbler “encrypts” the circuit (i.e., “garbles” the circuit)

Garbler can send garbled truth tables and keys for its inputs.

**Question:** how does evaluator obtain keys for its input?

Garbler can send garbled truth tables and keys for its inputs.
Step 3: Evaluator uses “oblivious transfer” to obtain keys for its input

For each wire corresponding to evaluator’s input, the garbler has two keys.

For each input wire, evaluator wants to obtain key corresponding to its input value.

At the end of the oblivious transfer protocol, garbler learns nothing about which key evaluator obtains, and evaluator learns exactly one of the two keys.
Yao’s Protocol for Two-Party Computation [Yao82]

Two-round protocol for secure two-party communication

OT message for keys corresponding to input wires

Keys communicated using OT (garbler does not know which keys are transmitted)

garbler

Evaluator uses keys to evaluate circuit gate-by-gate

Many improvements are possible to achieve better performance
Yao’s Protocol for Two-Party Computation [Yao82]

Two-round protocol for secure two-party communication

OT message for keys corresponding to input wires

Keys communicated using OT (garbler does not know which keys are transmitted)

Many improvements are possible to achieve better performance

Protocol is very efficient; communication is the bottleneck
General techniques apply to many different scenarios for diagnosing Mendelian diseases. Simple frequency-based filters are useful for rare disease diagnosis and can be efficiently evaluated in a privacy-preserving manner.
But What About More Complex Diseases?

Genome-wide association studies (GWAS):
- Identify genetic variants most correlated with a particular disease (or particular phenotype)
- Oftentimes, focused on identifying complex interactions between many variants

Control group (healthy)

Case group (affected)
But What About More Complex Diseases?

Each patient has a vector of SNPs (variations in specific locations in genome – 3 types)

<table>
<thead>
<tr>
<th>Patient</th>
<th>SNPs</th>
<th>Disease Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>0 1 0 0 2 2</td>
<td>0</td>
</tr>
<tr>
<td>Healthy</td>
<td>0 1 1 0 2 2</td>
<td>0</td>
</tr>
<tr>
<td>Healthy</td>
<td>1 1 0 0 2 2</td>
<td>0</td>
</tr>
<tr>
<td>Healthy</td>
<td>1 0 0 0 2 2</td>
<td>0</td>
</tr>
<tr>
<td>Healthy</td>
<td>2 1 1 2 2 2</td>
<td>1</td>
</tr>
<tr>
<td>Healthy</td>
<td>2 1 0 2 2 2</td>
<td>1</td>
</tr>
<tr>
<td>Healthy</td>
<td>2 1 0 2 2 1</td>
<td>1</td>
</tr>
<tr>
<td>Healthy</td>
<td>2 1 0 2 2 2</td>
<td>1</td>
</tr>
</tbody>
</table>

Goal: identify SNPs that are most correlated with disease status
But What About More Complex Diseases?

Each patient has a vector of SNPs (variations in specific locations in genome – 3 types)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>2</th>
<th>2</th>
<th>...</th>
<th>1</th>
</tr>
</thead>
</table>
| Healthy individuals
|        | 0  | 1  | 1  | 0  | 2  | 2  | ... | 1  |
|        | 1  | 1  | 0  | 0  | 2  | 2  | ... | 0  |
| Disease status
|        | 0  |    |    |    |    |    | 0   |    |
|        | 0  |    |    |    |    |    | 0   |    |
|        | 0  |    |    |    |    |    | 0   |    |

Goal: identify SNPs that are most correlated with disease status

Patients with lung cancer

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>0</th>
<th>...</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unlike Mendelian diseases, we are looking for many associations (e.g., several hundred)
But What About More Complex Diseases?

<table>
<thead>
<tr>
<th>Disease status</th>
<th>Healthy individuals</th>
<th>Patients with lung cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>≈ 25,000 individuals</td>
<td>Healthy individual 1</td>
<td>Patient 1</td>
</tr>
<tr>
<td>≈ 500,000 SNPs</td>
<td>Healthy individual 1</td>
<td>Patient 1</td>
</tr>
<tr>
<td>Healthy individual 2</td>
<td>Patient 2</td>
<td></td>
</tr>
<tr>
<td>Healthy individual 3</td>
<td>Patient 3</td>
<td></td>
</tr>
<tr>
<td>Healthy individual 4</td>
<td>Patient 4</td>
<td></td>
</tr>
</tbody>
</table>

Disease status

- Healthy individual 1: 0 1 0 0 2 2 2 ... 1
- Healthy individual 2: 0 1 1 0 2 2 2 ... 1
- Healthy individual 3: 1 1 0 0 2 2 2 ... 0
- Healthy individual 4: 2 1 1 2 2 0 ... 0
- Patient 1: 0
- Patient 2: 0
- Patient 3: 0
- Patient 4: 1

Cho-W-Berger [Nature Biotechnology 2018]
But What About More Complex Diseases?

---

Approximately 500,000 SNPs

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>2</th>
<th>2</th>
<th>...</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>3rd</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>4th</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Disease status

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0th</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1st</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3rd</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Approximately 25,000 individuals

Challenge: in real GWAS studies, we need to correct for population-level differences between groups.

---

Cho-W-Berger [Nature Biotechnology 2018]
GWAS computations most naturally expressed as arithmetic computations (e.g., matrix operations)

Recall: to apply Yao’s protocol, must first represent computation as a Boolean circuit

Can introduce significant overhead for arithmetic computations!
Arithmetic Computations on Shared Data

Patients “secret share” their data with two non-colluding hospitals

Approach: directly compute on secret-shared data
Arithmetic Computations on Shared Data

All operations done over a ring \((\mathbb{Z}_p)\)

\[
\begin{align*}
[v_1]_1 + [v_1]_2 &= v_1 \\
[v_2]_1 + [v_2]_2 &= v_2
\end{align*}
\]
Arithmetic Computations on Shared Data

Observation: each party can locally compute on their shares to obtain a share of the sum

\[ [v_1]_1 + [v_2]_1 = [v_1 + v_2]_1 \]

\[ [v_1]_2 + [v_2]_2 = [v_1 + v_2]_2 \]
For computing products on shared values (e.g., matrix-vector products, inner products, etc.), we can use a single-round interactive protocol [Bea91].
What About More Complex Diseases?

**This work:** first end-to-end GWAS protocol (with population correction)

- Based on computing on secret-shared inputs
- For 25K individuals, computation completes in about 3 days: feasible for performing large-scale scientific studies

**Approach:** directly compute on secret-shared data
Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes.
Modern cryptographic tools enable useful computations while protecting the privacy of individual genomes.
Secure Genome Computation

MPC Protocol

Project Website:
https://crypto.stanford.edu/~dwu4/genomepriv-project.html

Thank you!