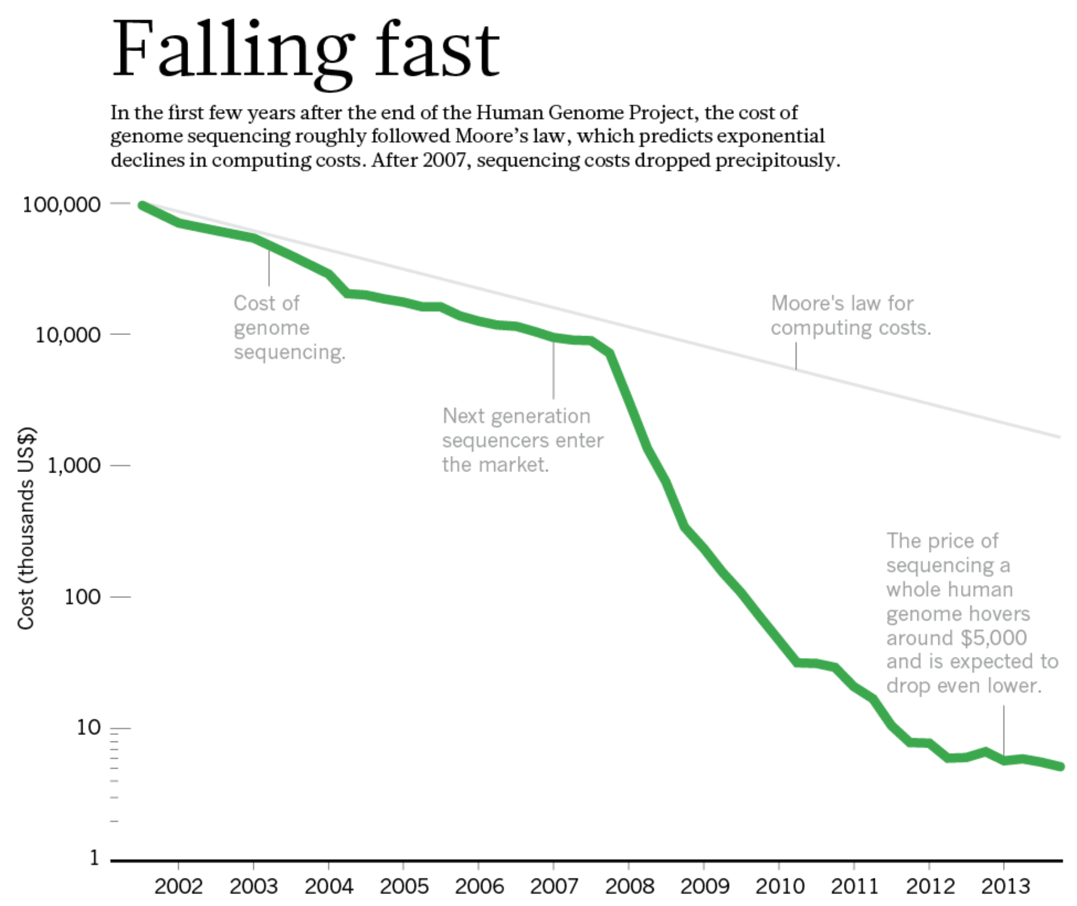
Privacy-Preserving Genome Analysis

David Wu

Joint work with Hyunghoon Cho and Dan Boneh

The Cost of DNA Sequencing

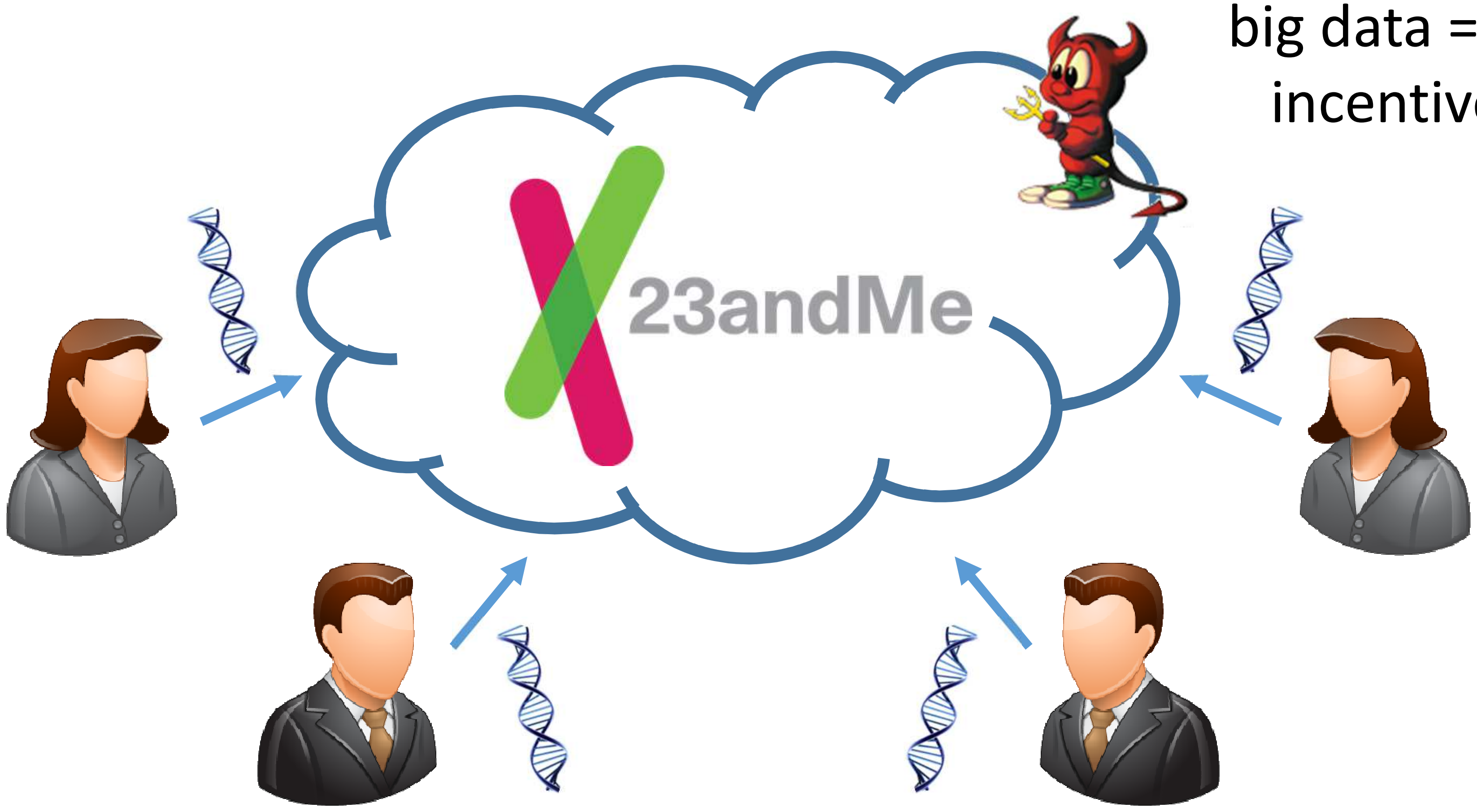


Genome sequencing around $1000!

Source: *Nature*, 2014

The Era of Personal Genomics

big data = big incentives



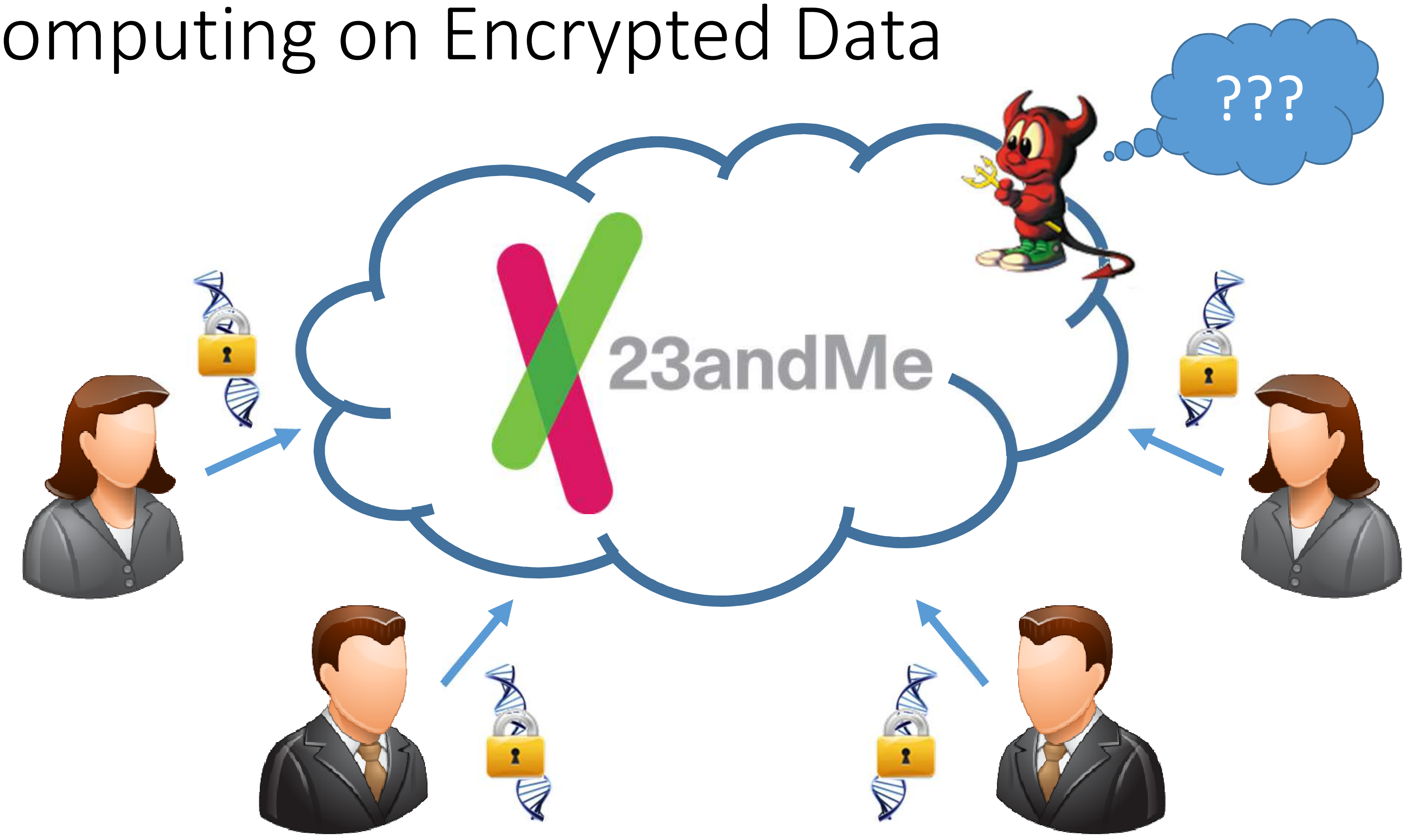
The Era of Personal Genomics

Can we compute on our genomes without sacrificing our personal privacy?

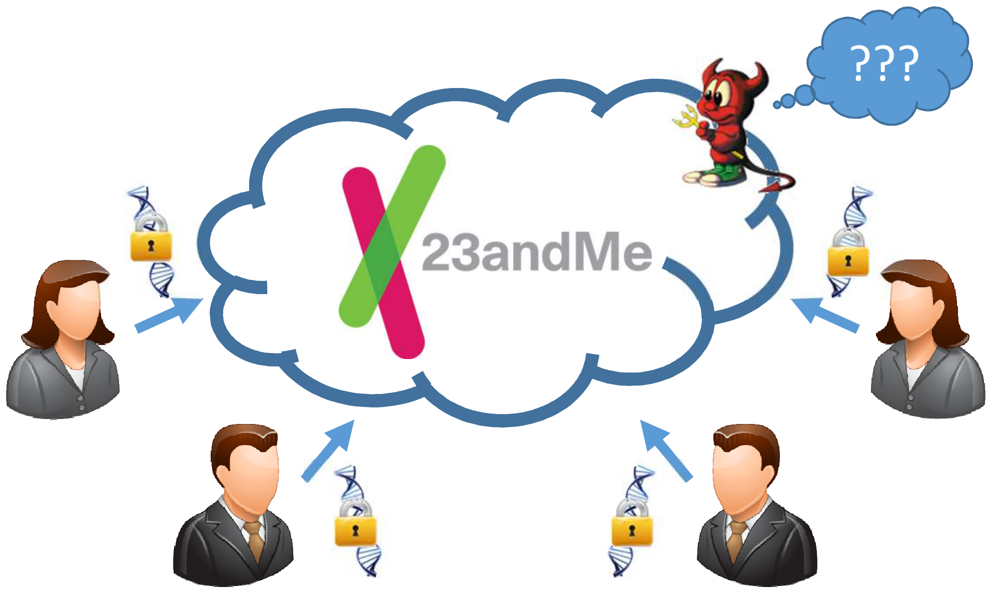
Computing on Encrypted Data

Why not simply encrypt our genomes?

Computing on Encrypted Data



Computing on Encrypted Data

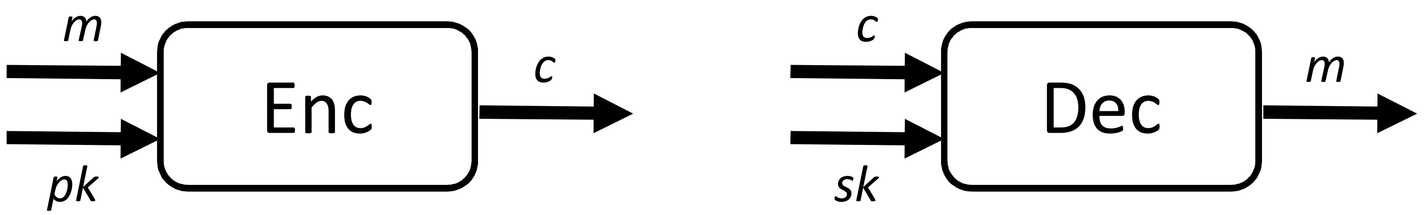


But if adversary cannot learn from the data, then neither can the cloud!

Homomorphic Encryption

Homomorphic encryption (HE): encryption schemes that support computation on ciphertexts

Consists of three functions:

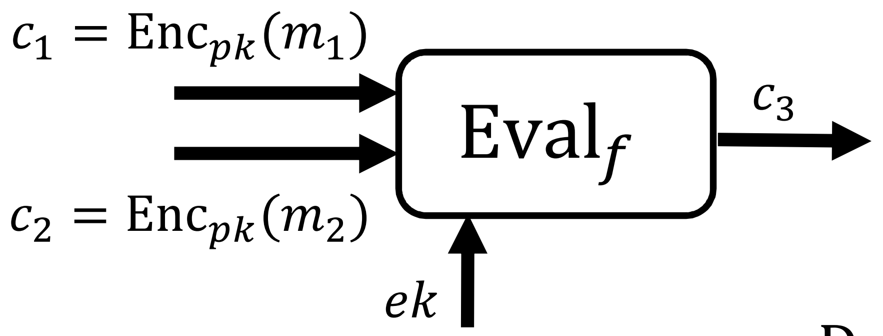


Must satisfy usual notion of semantic security

Homomorphic Encryption

Homomorphic encryption: encryption schemes that support computation on ciphertexts

Consists of three functions:



$c_1 = \mathrm{Enc}_{pk}(m_1)$
$c_2 = \mathrm{Enc}_{pk}(m_2)$
Eval_f
c_3
ek

$$\mathrm{Dec}_{sk}\left(\mathrm{Eval}_f(ek, c_1, c_2)\right) = f(m_1, m_2)$$

Fully Homomorphic Encryption (FHE)

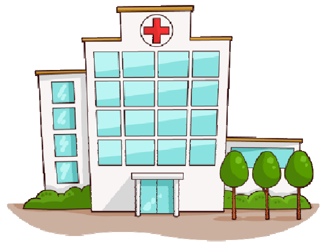
Many homomorphic encryption schemes:

- ElGamal: $f(m_0, m_1) = m_0 m_1$
- Paillier: $f(m_0, m_1) = m_0 + m_1$

Fully homomorphic encryption: homomorphic with respect to **two** operations: addition and multiplication

- [BGN05]: one multiplication, many additions (SWHE)
- [Gen09]: first FHE construction from lattices

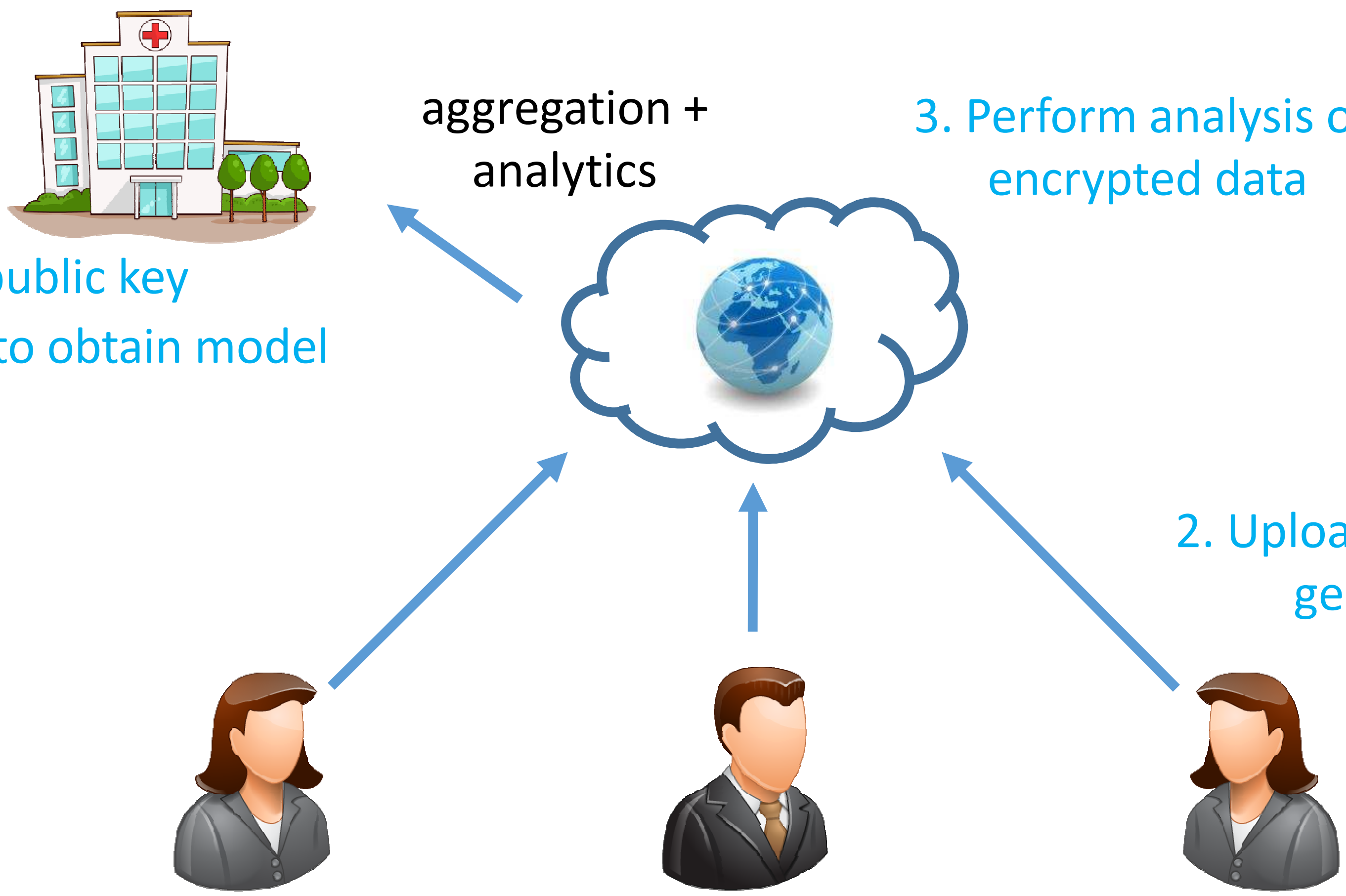
Outsourcing via FHE (Hypothetical)



Suppose a medical institution wants to do a population-wide study using genomic data

Individuals might want to participate, but reluctant to simply share their genome

Outsourcing via FHE (Hypothetical)



The iDASH 2015 Competition

A competition to explore the viability of homomorphic encryption (and multiparty computation) for secure genomic analysis

Two tasks:

- Secure outsourcing of GWAS statistics
- Computing Hamming distance between two sequences

Task 1: Computing GWAS Statistics

Genome-wide association study (GWAS): finding associations between single-nucleotide polymorphisms (SNPs) and traits (e.g., certain diseases)

Case:	`AA`	`AG`	`AA`	`AG`	`GG`
Control:	`AG`	`AG`	`GA`	`GG`	`GG`

Genotypes for different individuals at a fixed location in the genome

Two different metrics of interest: minor allele frequency (MAF) and χ^2 statistic

Task 1: Computing GWAS Statistics

Case:	AA	AG	AA	AG	GG
Control:	AG	AG	GA	GG	GG

Genotypes for different individuals at a fixed location in the genome

allele counts

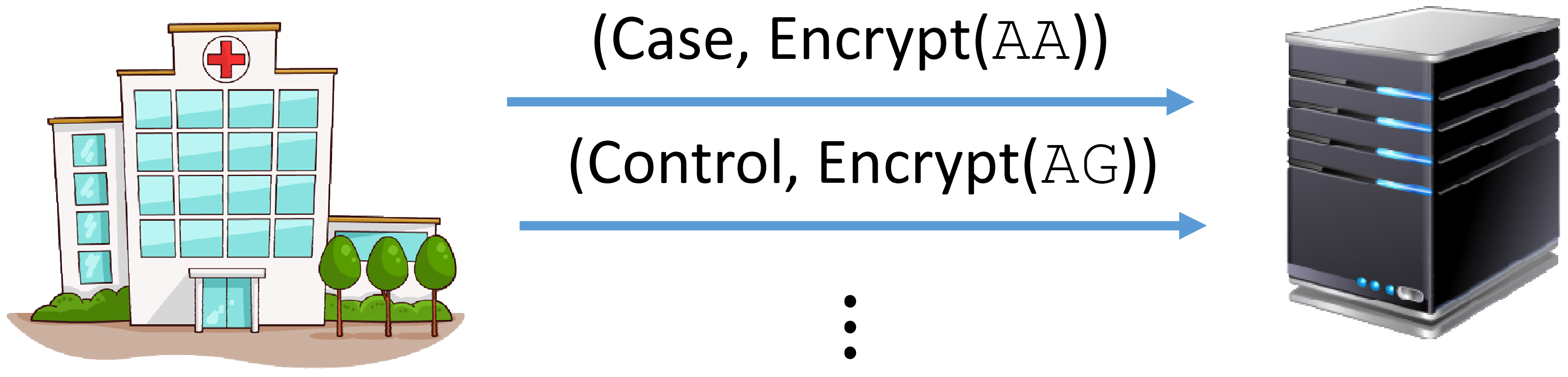
Minor Allele Frequency: $\frac{\min(n_A, n_G)}{n_A + n_G}$

χ^2-statistic: $\chi^2 = \sum \frac{(\mathrm{Obs} - \mathrm{Exp})^2}{\mathrm{Exp}}$

Observed (Obs) and expected (Exp) are functions of the different allele counts in the case and control groups

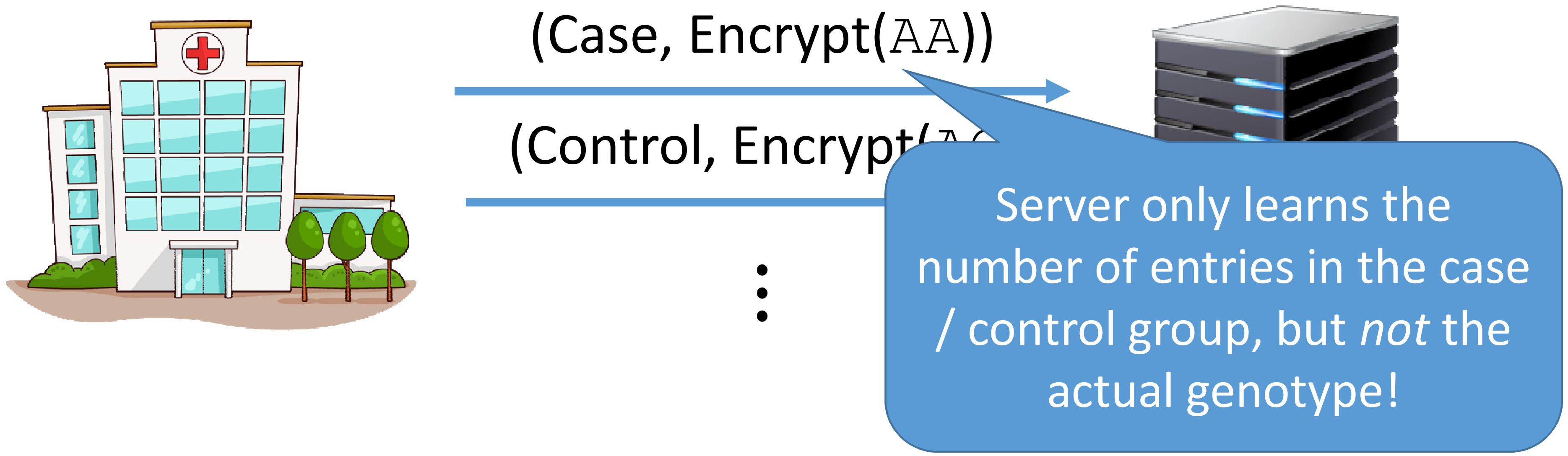
Task 1: Computing GWAS Statistics

Setting: hospital or medical institution sequences patients’ genomes and stores the data encrypted



Task 1: Computing GWAS Statistics

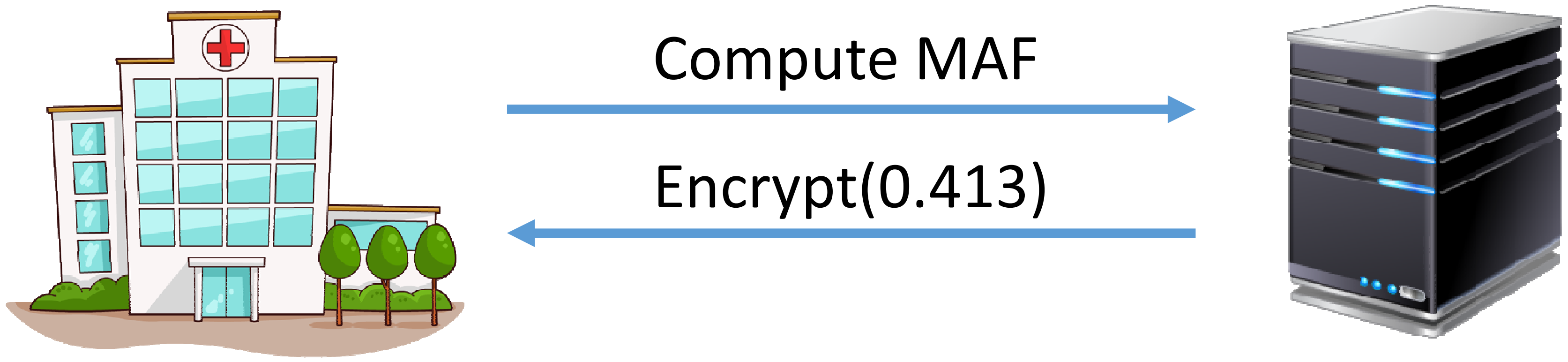
Setting: hospital or medical institution sequences patients' genomes and stores the data encrypted



Server only learns the number of entries in the case / control group, but *not* the actual genotype!

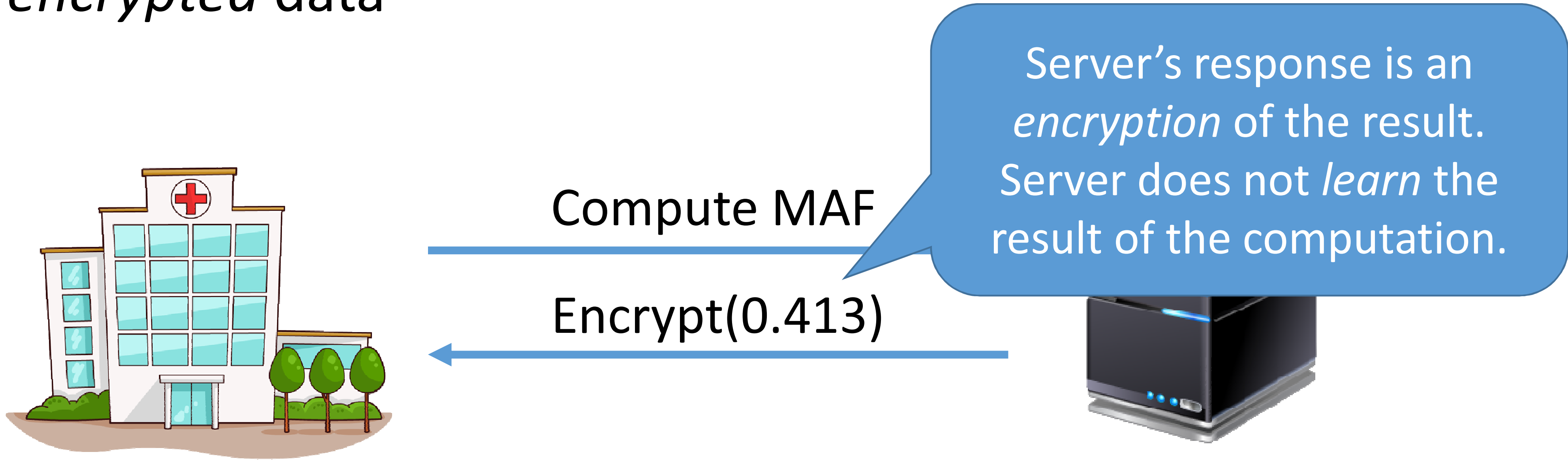
Task 1: Computing GWAS Statistics

Want server to be able to compute GWAS statistics on *encrypted* data



Task 1: Computing GWAS Statistics

Want server to be able to compute GWAS statistics on *encrypted* data


Compute MAF
Encrypt(0.413)
Server's response is an *encryption* of the result. Server does not *learn* the result of the computation.

Striking a Balance

Minor Allele Frequency: $\frac{\min(n_A, n_G)}{n_A + n_G}$

χ^2-statistic: $\chi^2 = \sum \frac{(\mathrm{Obs} - \mathrm{Exp})^2}{\mathrm{Exp}}$

Observation: allele counts are sufficient for computing MAF and χ^2

Solution: delegate *aggregation* to the cloud, client computes the statistical quantities of interest

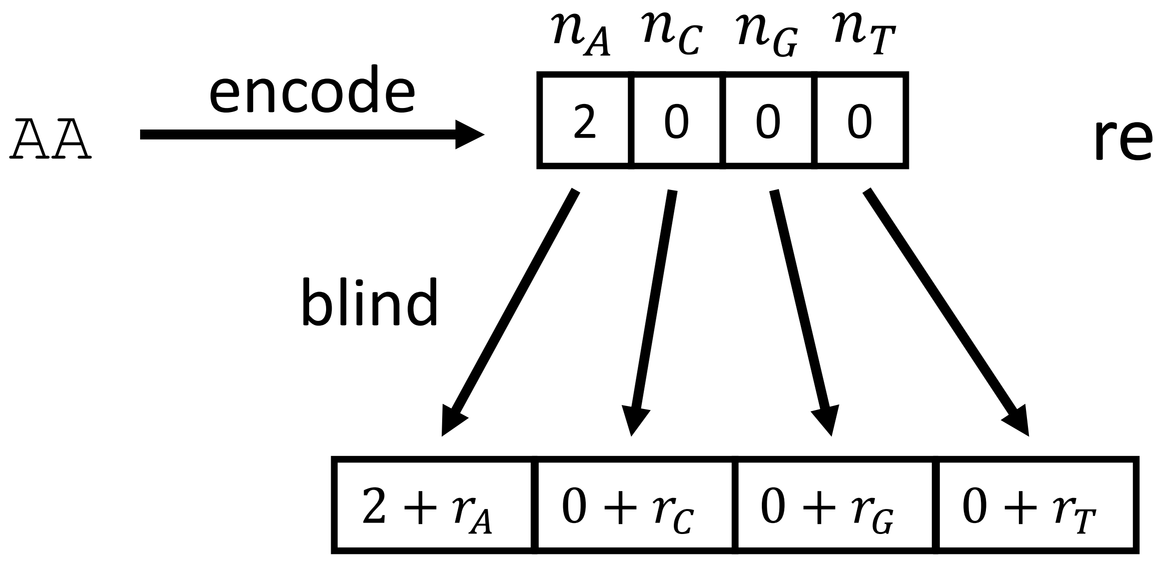
Practical Outsourcing

Solution: delegate *aggregation* to the cloud, client computes the statistical quantities of interest

Solution enables use of symmetric primitives (e.g., AES)

Symmetric primitives + arithmetic faster than public key decryption

Symmetric Encryption



each genotype represented as a vector of counts

encrypt entries by adding independent, blinding factors from $\mathbb{Z}_n$

Symmetric Encryption

$$\texttt{AA} \longrightarrow \boxed{2 + r_A}\boxed{0 + r_C}\boxed{0 + r_G}\boxed{0 + r_T}$$

$$\texttt{AG} \longrightarrow \boxed{1 + r'_A}\boxed{0 + r'_C}\boxed{1 + r'_G}\boxed{0 + r'_T}$$

$$\text{Sum} \longrightarrow \boxed{3 + r_A + r'_A}\boxed{0 + r_c + r'_C}\boxed{1 + r_G + r'_G}\boxed{0 + r_T + r'_T}$$

decryption: compute blinding factors and subtract

Symmetric Encryption

generate blinding factors using
$\mathrm{PRF}(k, \mathrm{tag})$

tag: `SNP id` $\|$ `group id` $\|$ `subject id`

$$\texttt{AA} \longrightarrow \boxed{2 + r_A}\boxed{0 + r_C}\boxed{0 + r_G}\boxed{0 + r_T}$$

Symmetric Encryption

Homomorphic operations consist of only **additions**

Encryption and decryption are **symmetric** primitives

Further Improvements

Client must do linear work to decrypt

- Alternative: if the data comes in batches, the client can precompute the counts per batch during encryption
- Decryption time proportional to *number of batches*

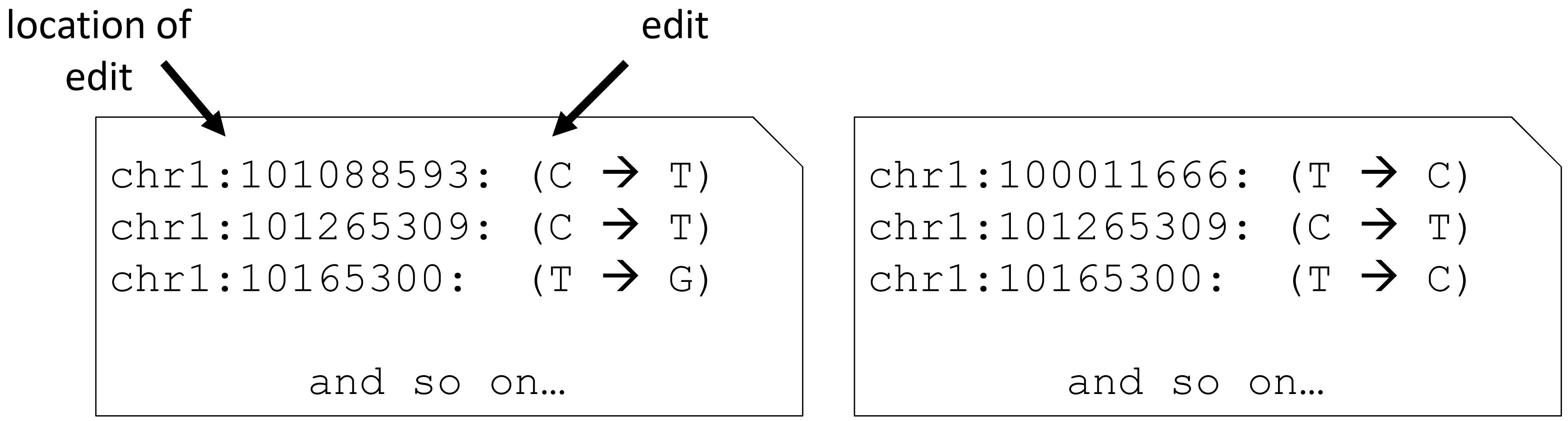
Performance

Timing (in seconds) for computing MAF + χ^2 statistics (500 subjects)

# SNPs	Encryption	Aggregation	Decryption
100	0.17	0.02	0.15
1,000	1.68	0.17	1.42
10,000	17.47	1.59	15.06
100,000	179.53	17.72	145.52

Only a few hundred lines of C++ code to implement!

Task 2: Hamming Distance Computation



```
chr1:101088593: (C → T)
chr1:101265309: (C → T)
chr1:10165300:  (T → G)

         and so on…
```

```
chr1:100011666: (T → C)
chr1:101265309: (C → T)
chr1:10165300:  (T → C)

         and so on…
```

compute the Hamming distance between two sequences (represented as edits with respect to a reference genome)

Task 2: Hamming Distance Computation

```
chr1:101088593: (C → T)
chr1:101265309: (C → T)
chr1:10165300:  (T → G)

      and so on…
```

→ ATGCTTAGTGGC…

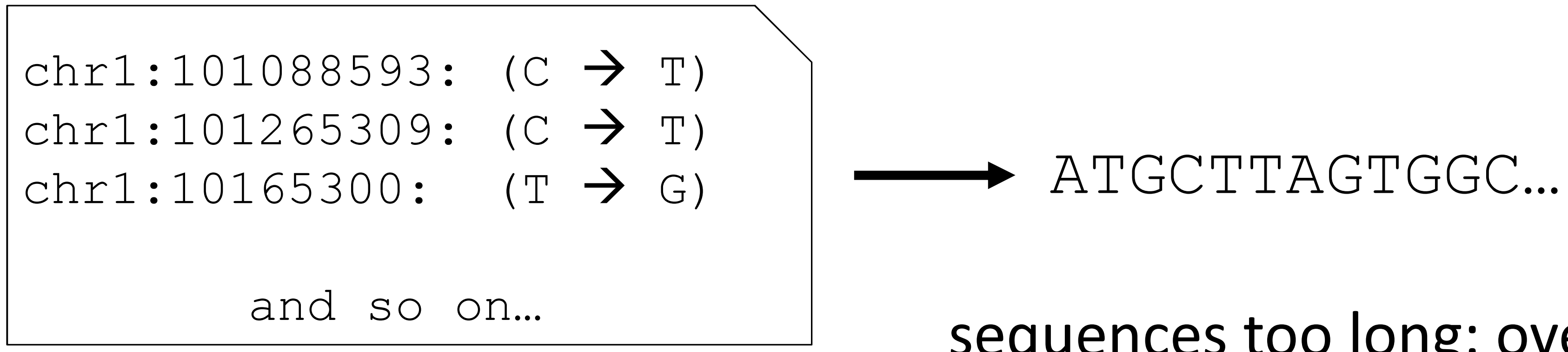
```
chr1:100011666: (T → C)
chr1:101265309: (C → T)
chr1:10165300:  (T → C)

      and so on…
```

→ ACGCTTGGTGGC…

naïve method: expand sequences, pairwise equality test

Task 2: Hamming Distance Computation



```
chr1:101088593: (C → T)
chr1:101265309: (C → T)
chr1:10165300:  (T → G)

       and so on…
```

sequences too long: over 3 billion base pairs in human genome

desire: protocol with performance proportional to *number of edits*

Task 2: Hamming Distance Computation

```
chr1:101088593: (C → T)
chr1:101265309: (C → T)
chr1:10165300:  (T → G)

         and so on…
```

Genome A

```
chr1:100011666: (T → C)
chr1:101265309: (C → T)
chr1:10165300:  (T → C)

         and so on…
```

Genome B

view genomes as sets of edits from reference:

$$d_H(A, B) = |A| + |B| - 2 \cdot |A \cap B|$$

Homomorphic Set Intersection

```
chr1:101088593: (C → T)
chr1:101265309: (C → T)
chr1:10165300:  (T → G)

        and so on…
```

```
chr1:100011666: (T → C)
chr1:101265309: (C → T)
chr1:10165300:  (T → C)

        and so on…
```

Equality function: $f(x, y) = \mathbf{1}\{x = y\}$

Simple solution: sum over pairwise equality tests

Homomorphic Set Intersection

Homomorphic evaluation of equality function:

If $x, y \in \{0,1\}$,

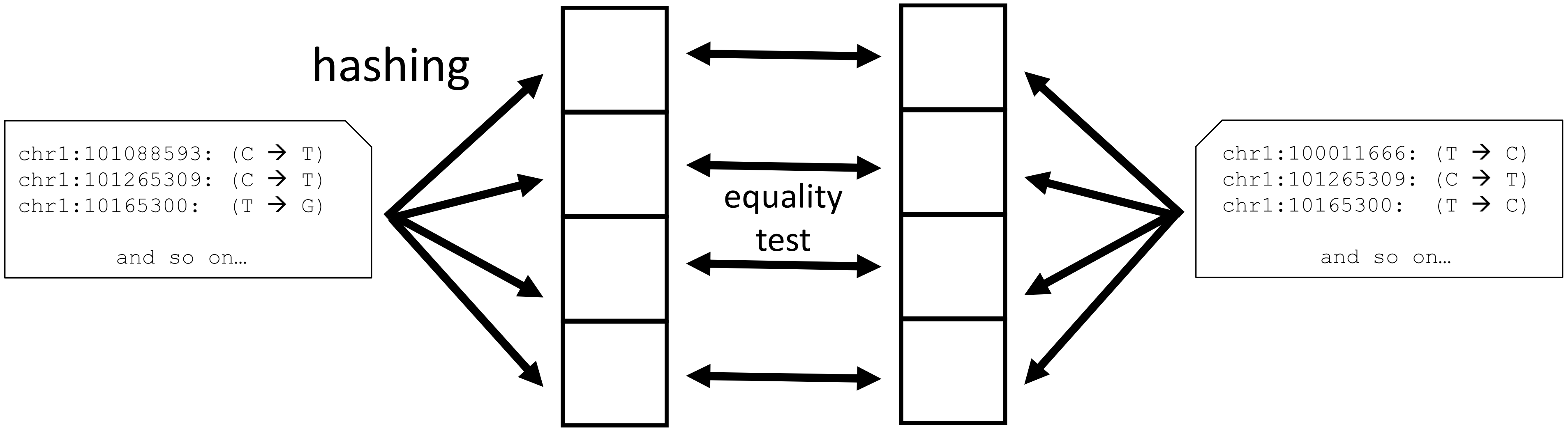
$$f(x,y) = \mathbf{1}\{x = y\} = 1 - (x - y)^2$$

Easy to generalize to n bit integers, but requires degree $2n$ homomorphism

requires somewhat homomorphic encryption

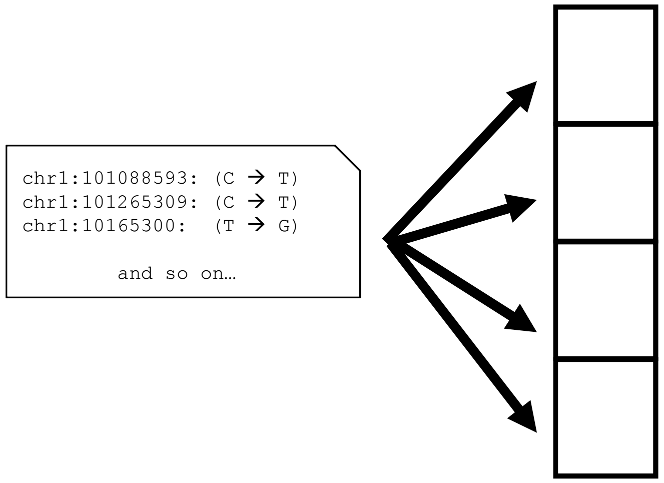
Homomorphic Set Intersection

Hashing to decrease number of pairwise comparisons



hash elements into buckets, pairwise equality test on hashed values within buckets

Homomorphic Set Intersection: Tradeoffs



Tunable parameters:

- number of buckets
- bits used to represent each element in a bucket
- bucket size

More buckets → lower collision rate, possibly more ciphertexts

More bits → lower collision rate, more homomorphism for equality test

Larger buckets → less likely that bucket overflows

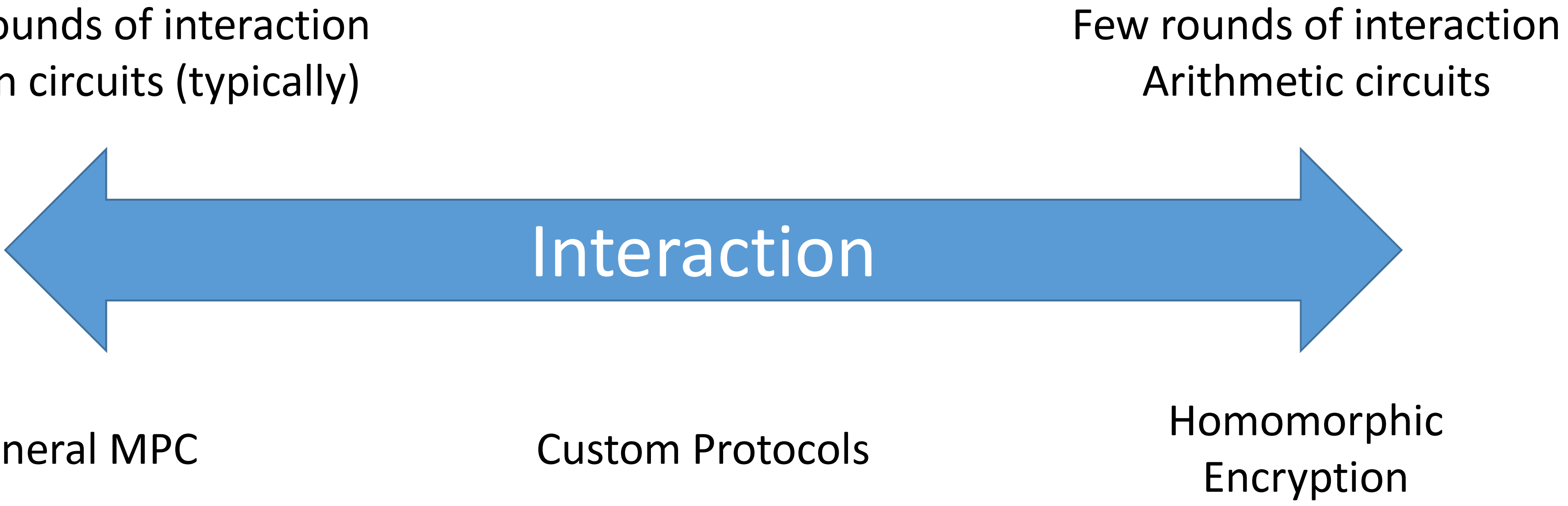
Performance

Timing (in seconds) for homomorphic set intersection using HELib:

Size of Sets	Key Generation	Hashing	Encryption	Computation	Encryption
1,000	23.80	0.007	31.97	104.16	1.78
5,000	23.36	0.025	95.38	475.37	1.78
10,000	27.14	0.093	176.50	936.64	1.91

Primary drawback: key sizes + ciphertext sizes very large (several hundred MB to just over 1 GB)

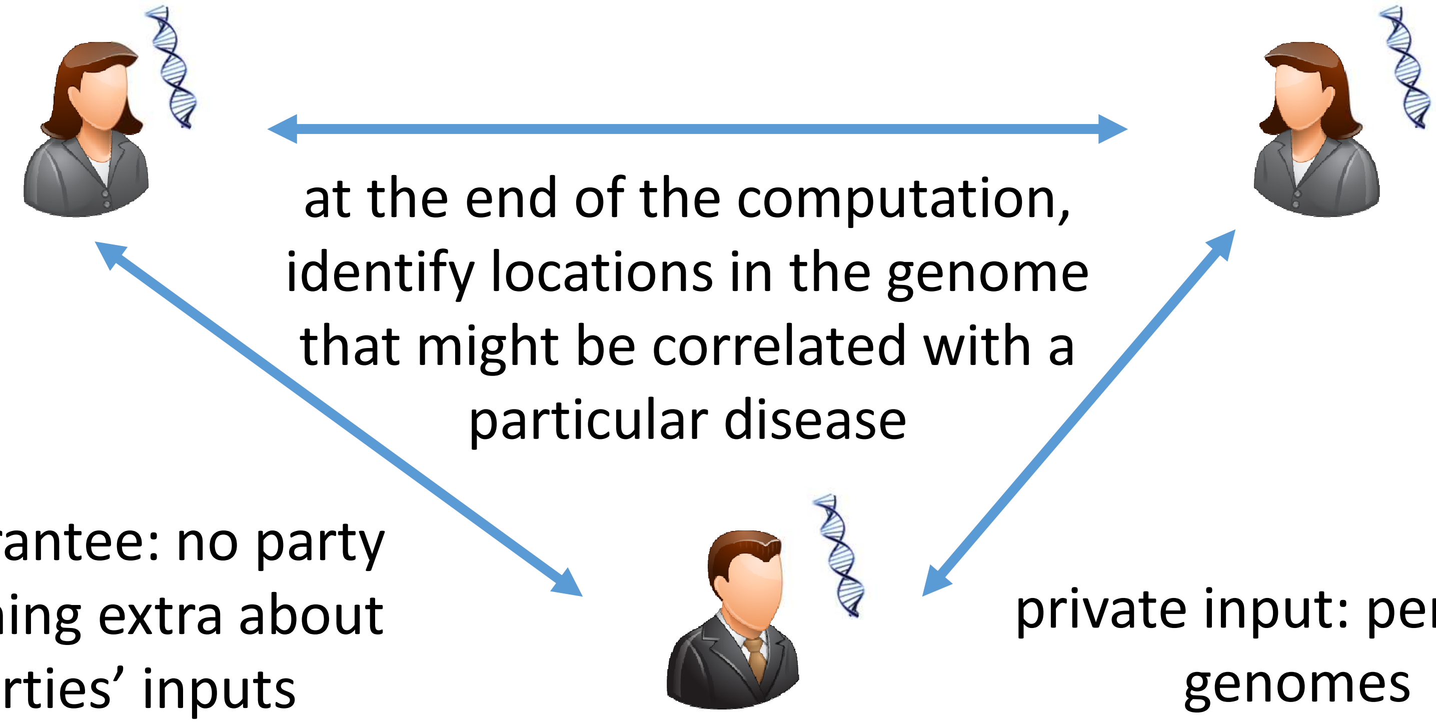
The Other Side of the Spectrum


Many rounds of interaction
Boolean circuits (typically)
Few rounds of interaction
Arithmetic circuits
Interaction
General MPC
Custom Protocols
Homomorphic Encryption

General methods for secure computation

Secure Multiparty Computation (MPC)

Multiple parties want to compute a joint function on *private* inputs



at the end of the computation, identify locations in the genome that might be correlated with a particular disease

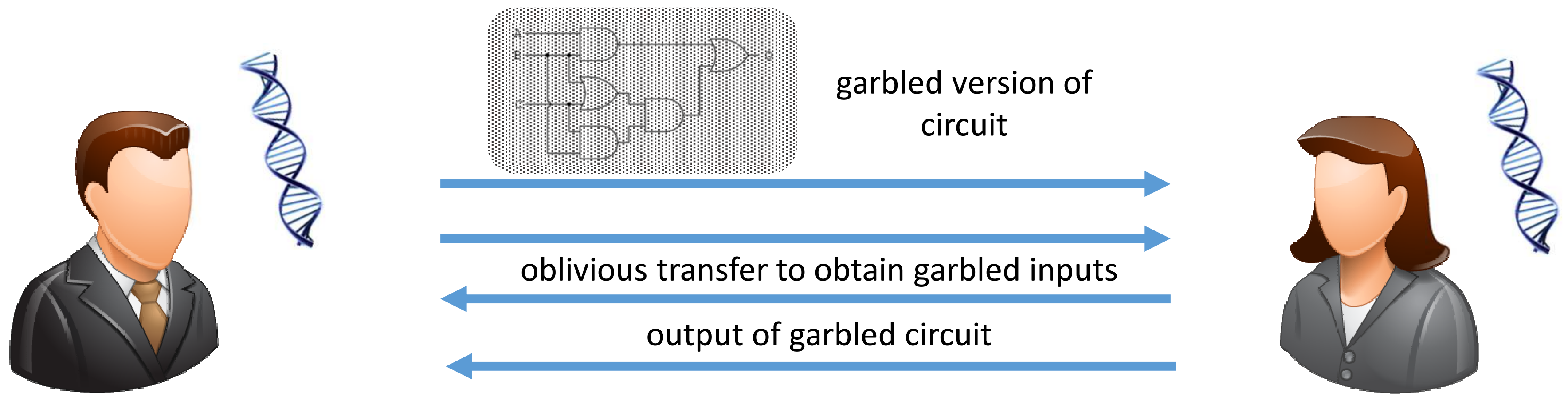
privacy guarantee: no party learns anything extra about other parties' inputs

private input: personal genomes

Two Party Computation (2PC)

- Simpler scenario: two-party computation (2PC)

- 2PC: Mostly "solved" problem: Yao's circuits [Yao82]
  - Express function as a Boolean circuit


garbled version of circuit
oblivious transfer to obtain garbled inputs
output of garbled circuit

Two-Party Computation (2PC)

- Yao’s circuits very efficient and heavily optimized [KSS09]
    - Evaluating circuits with 1.29 ***billion*** gates in 18 minutes (1.2 gates / µs) [ALSZ13]

Secure Multiparty Computation

- General MPC suffices to evaluate arbitrary functions amongst many parties: should be viewed as a <u>feasibility</u> result
- Limitations of general MPC
    - more rounds of communication / interaction
    - possibly large bandwidth
    - hard to coordinate interactions with large number of parties

Concluding Remarks

- Personal genomics introduces many new opportunities, but also many new security and privacy risks
- Many existing cryptographic tools exist for *private* and *secure* computation
  - SWHE / FHE: non-interactive computation on encrypted data
  - 2PC / MPC: interactive computation on private inputs
  - Future: hardware support for secure computation?
- Many different tradeoffs in terms of communication, computation, rounds of interaction

Thanks!