# PRESAGE: PRIVACY-PRESERVING GENETIC TESTING VIA INTEL SGX

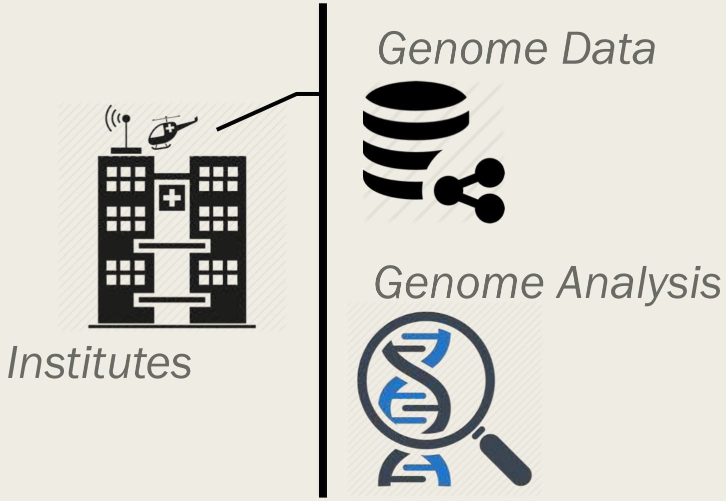
S. C. SAHINALP

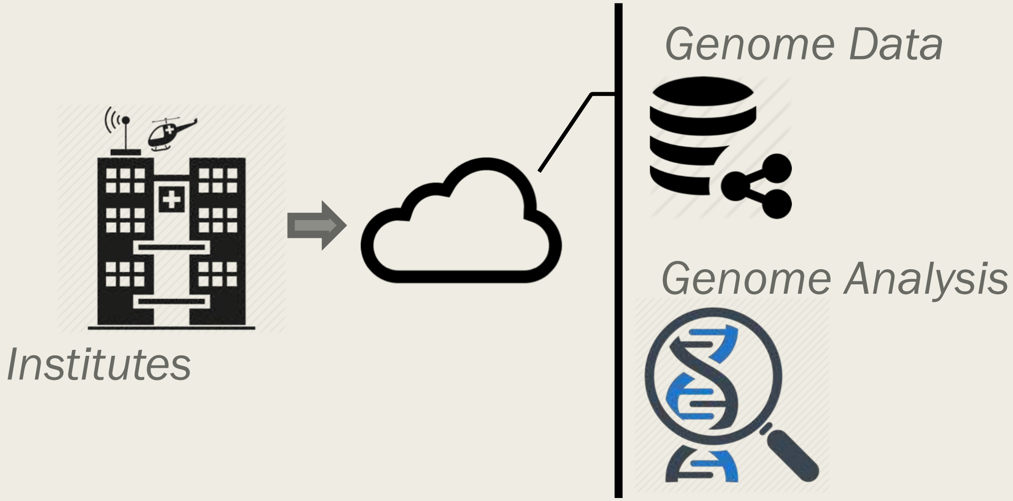
BOGAZ’DA YAZ OKULU 2018

# Motivation

- It is becoming a big challenge to efficiently store and process the huge amount of genomic data for individual biomedical research institutions.

**Cloud Computing** emerges as an ideal platform for providing elastic computation and storage resources for genomic data analysis





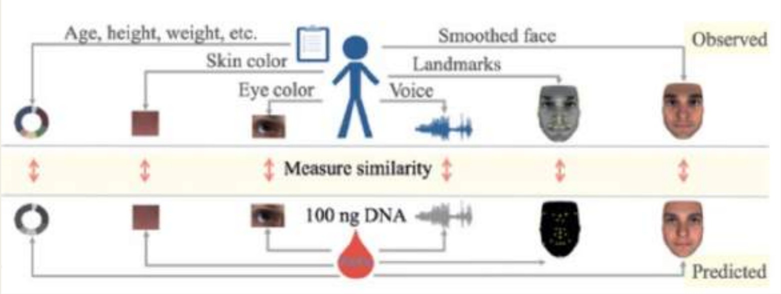
# Motivation

- Individual genomic information tends to reveal sensitive personal information. Thus privacy concerns have posed challenges to outsource genomic data in an untrusted cloud environment

•***Lin et. al. 2004 Science***: 75 or more SNPs (Single-nucleotide polymorphism) will be sufficient to identify a single person.

•***Gymrek et al. 2013 Science***: surnames can be recovered from personal genomes, linking Utah Residents with Northern and Western European Ancestry (CEU) and public genetic genealogy databases (Ysearch & SMGF).

•***Lipper et.al. 2017 PNAS***: Prediction of human physical traits and demographic information.



***Homer et al. 2008 PLOS Genetics:*** aggregated genome data (i.e., allele frequencies) can also be used for re-identifying an individual in a case group with a certain disease.

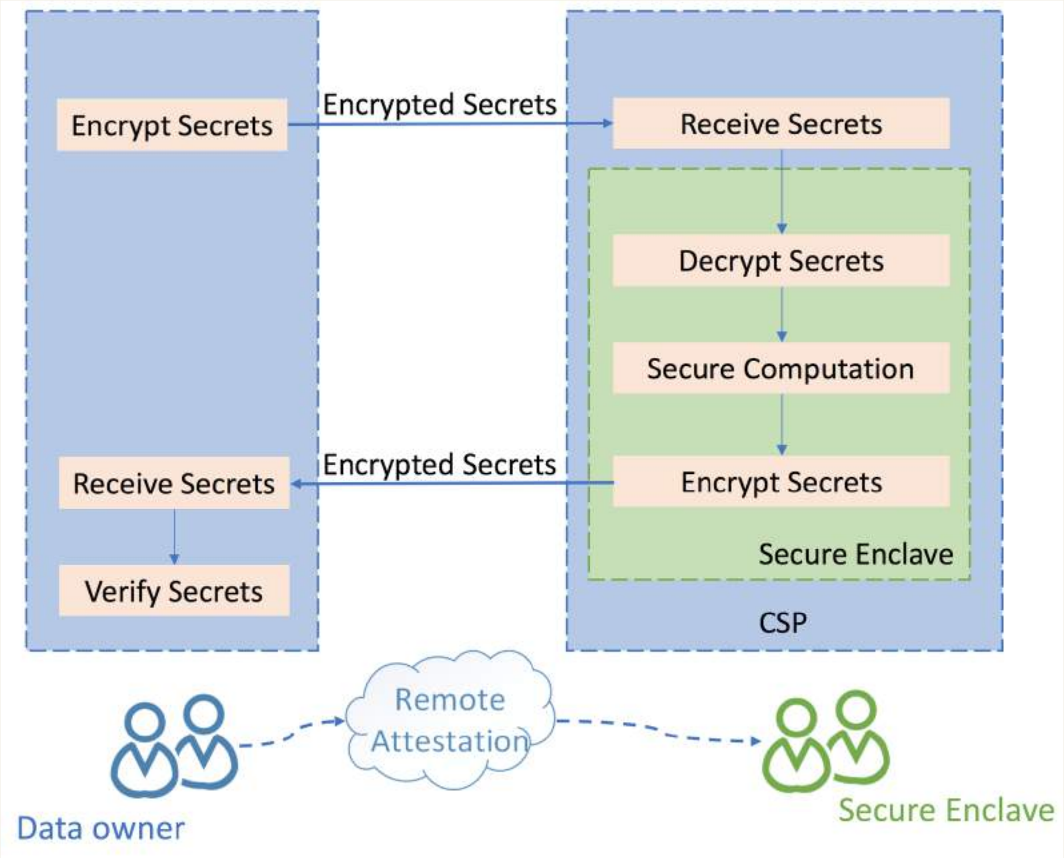
# Our Solution

- We present one of the first implementations of SGX based secure genetic testing framework to facilitate efficiently outsourced storage and computation.

The secure outsource storage is achieved through data sealing scheme within SGX framework, which is immune to replay attack.

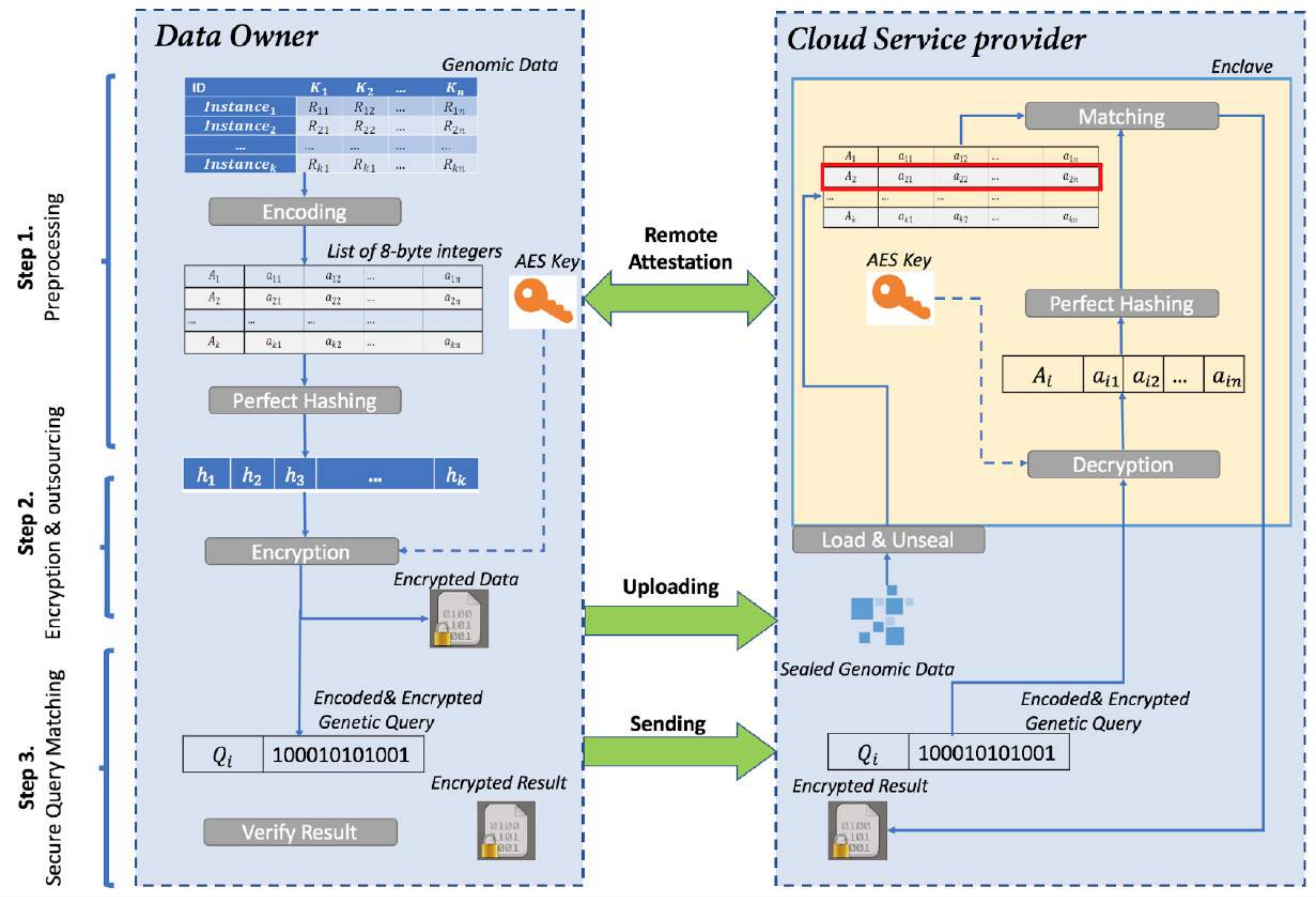
We have taken into account the oblivious access protection by using 4KB page-wise data access model.

To improve the performance, we adopt a perfect hashing scheme to achieve O(1) complexity data access within each 4KB page.



Intel SGX Based Secure Genetic Testing Cloud

# Our Solution



| ID | $K_1$ | $K_2$ | ... | $K_n$ |
|---|---|---|---|---|
| $Instance_1$ | $R_{11}$ | $R_{12}$ | ... | $R_{1n}$ |
| $Instance_2$ | $R_{21}$ | $R_{22}$ | ... | $R_{2n}$ |
| ... | ... | ... | ... | ... |
| $Instance_k$ | $R_{k1}$ | $R_{k1}$ | ... | $R_{kn}$ |

| | | | | |
|---|---|---|---|---|
| $A_1$ | $a_{11}$ | $a_{12}$ | ... | $a_{1n}$ |
| $A_2$ | $a_{21}$ | $a_{22}$ | ... | $a_{2n}$ |
| ... | ... | ... | ... | |
| $A_k$ | $a_{k1}$ | $a_{k2}$ | ... | $a_{kn}$ |

Workflows of the proposed PRESAGE framework, presented in three consecutive steps:

1. **Preprocessing**
2. **Encryption and data outsourcing.**
3. **Secure Genetic Query Matching.**

# Experimental Studies

- *Dataset*: The dataset is presented in VCF format. And sizes of VCF datasets used in our experiments vary from **10,000** to **200,000** records.
- *Experiment Environment:* All of the experiments except the iDASH competition results are conducted on a Windows 10 SGX-enabled machine with **i7 6820HK CPU** and **48 GB** memory. Both data owner and CSP were simulated on the aforementioned SGX machine. The iDASH competition results were evaluated on the Linux server with an Xeon Processor **E3-1275 v5** and **64 GB** memory

# Results

Comparison of querying performance among PRESAGE, HME-based method and plaintext implementation

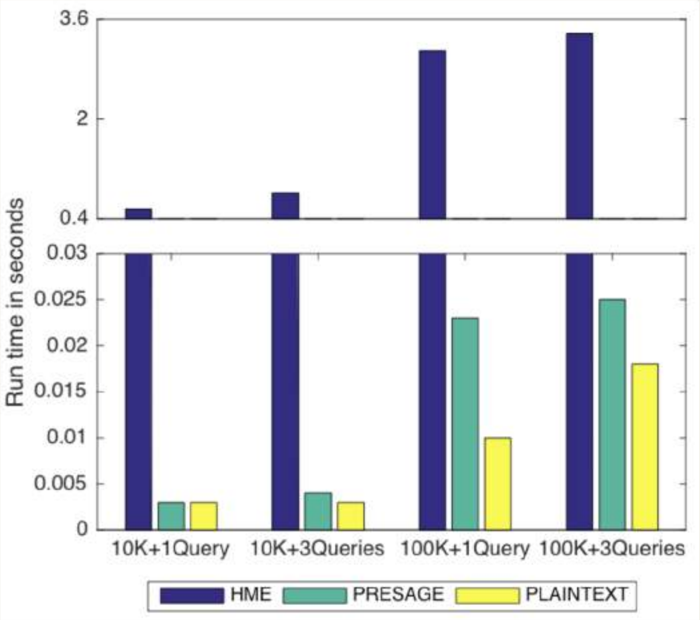


Table 1. The breakdown run time (in seconds) of the proposed PRESAGE framework

| Record size | Plaintext | Encoded data | Sealed data | Enclave memory usage | |
|---|---|---|---|---|---|
| | | | | 1 query | 3 queries |
| 10,000 | 0.55 MB | 0.09 MB | 0.12 MB | 3.006 MB | 3.006 MB |
| 50,000 | 2.59 MB | 0.45 MB | 0.59 MB | 3.010 MB | 3.010 MB |
| 100,000 | 5.26 MB | 0.90 MB | 1.15 MB | 3.010 MB | 3.010 MB |
| 200,000 | 10.5 MB | 1.75 MB | 2.31 MB | 3.010 MB | 3.010 MB |

Table 2. The data size and enclave memory consumption (in MB) for different datasets.

| Record size | Attestation | SNPs coding | Hash generation | Enclave creation | Data sealing | Number of queries | |
|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 3 |
| 10,000 | 0.121s | 0.016s | 1.130s | 0.169s | 0.094s | 0.003s | 0.003s |
| 50,000 | 0.126s | 0.080s | 6.371s | 0.173s | 0.517s | 0.012s | 0.013s |
| 100,000 | 0.124s | 0.164s | 13.473s | 0.179s | 0.980s | 0.023s | 0.025s |
| 200,000 | 0.120s | 0.309s | 28.677s | 0.171s | 2.045s | 0.043s | 0.048s |

# Conclusion

- We proposed a secure outsourcing framework, which can defend malicious attack. To improve the efficiency, an minimal perfect hashing scheme has been incorporated

- Our experiment results demonstrated the efficiency of the proposed PRESAGE framework. For a VCF file with 200K records, the PRESAGE securely processes a query within 0.05 seconds, which includes file loading, unsealing and query matching. Compared with state-of-the-art HME solution, PRESAGE framework shows at least 120x performance gain.

# Acknowledgements

- Feng Chen,
- Chenghong Wang,
- Wenrui Dai,
- Xiaoqian Jiang,
- Noman Mohammed,
- Md Momin Al Aziz,
- Md Nazmus Sadat,
- Kristin Lauter,
- Shuang Wang

# Secure GWAS via Intel SGX

CAN KOCKAN

# Motivation / Goal

- Enable secure whole genome variant search among multiple individuals from multiple institutions

- Institution A has VCF files from $x_A$ individuals labeled case/control, Institution B has VCF files from $x_B$ individuals labeled case/control, …

- Institutes don't want to share data, want to do GWAS on untrusted cloud

# Requirements

- Secure (Everything kept encrypted outside the SGX Enclave)
- Fast
- Accurate
- Scalable

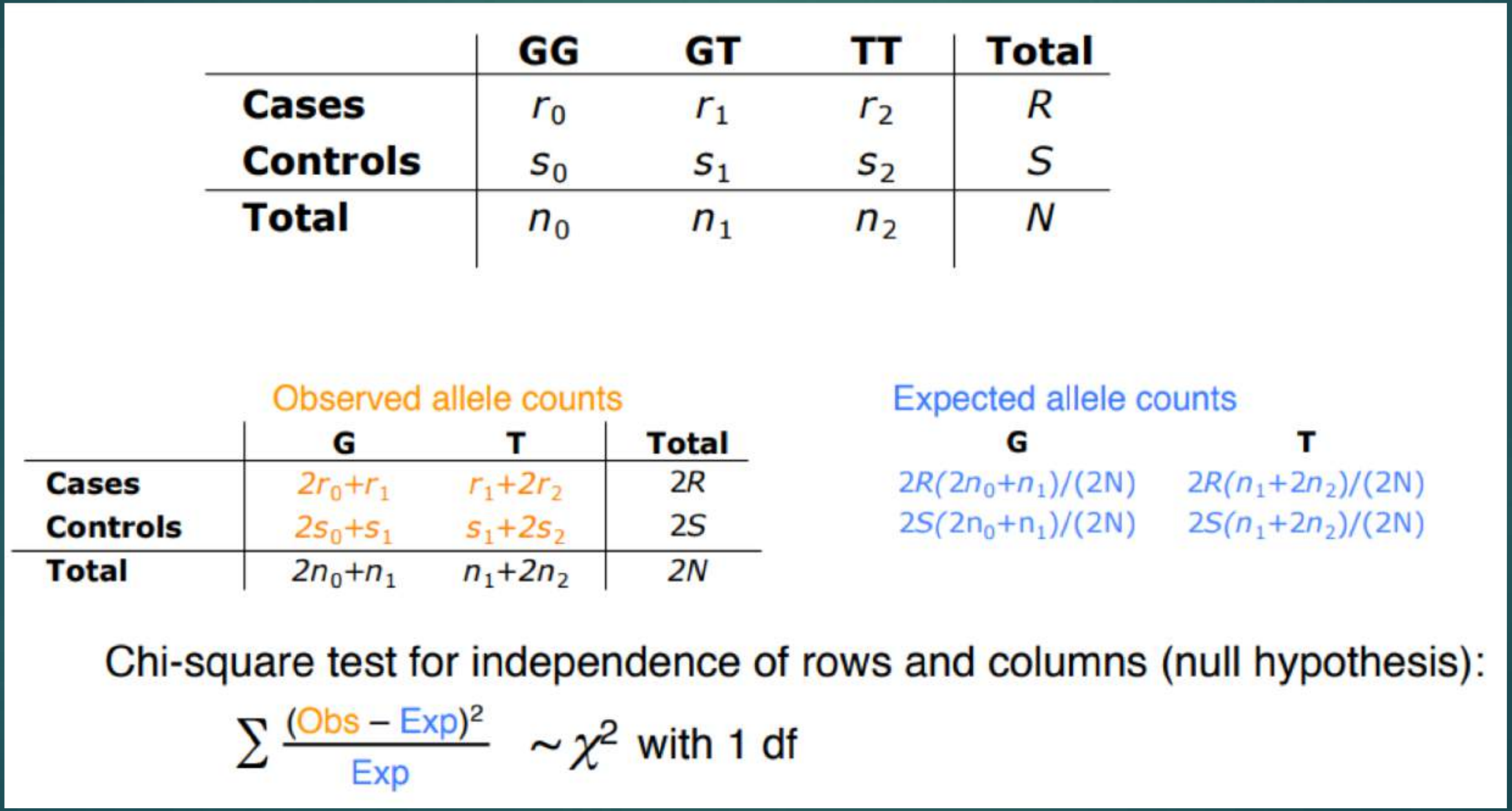
# Challenges

- Data is too large (iDASH dataset ~30GB, real-life much bigger)
- SGX Enclave max size 128 MB
- Linux allows 4GB paging – paging is extremely slow, typically many orders of magnitude slower than RAM
- More data longer transfer, more data slower encryption/decryption

# Solution Outline

- Keep hash table inside the SGX enclave (server)
- Filter and compress VCF files (Client(s))
- Construct Enclave (Server)
- Perform Remote Attestation (Both Parties Exchange Messages)
- Receive Data, Update Hash Table (Server)
- Calculate Top-K (K=10 for iDASH) SNPs wrt $X^2$ test (Server)

# Allele Counting for $X^2$ Test



| | GG | GT | TT | Total |
|---|---|---|---|---|
| **Cases** | $r_0$ | $r_1$ | $r_2$ | $R$ |
| **Controls** | $s_0$ | $s_1$ | $s_2$ | $S$ |
| **Total** | $n_0$ | $n_1$ | $n_2$ | $N$ |

Observed allele counts

| | G | T | Total |
|---|---|---|---|
| **Cases** | $2r_0+r_1$ | $r_1+2r_2$ | $2R$ |
| **Controls** | $2s_0+s_1$ | $s_1+2s_2$ | $2S$ |
| **Total** | $2n_0+n_1$ | $n_1+2n_2$ | $2N$ |

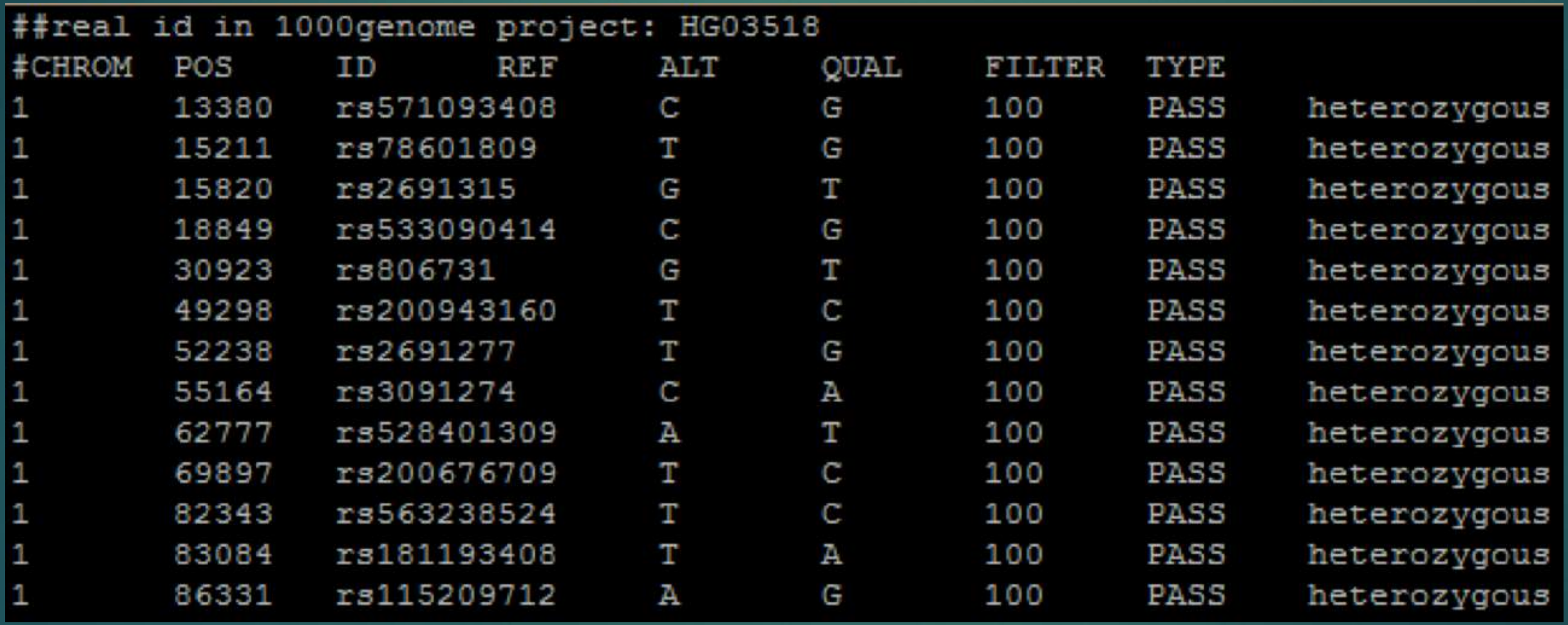
Expected allele counts

| G | T |
|---|---|
| $2R(2n_0+n_1)/(2N)$ | $2R(n_1+2n_2)/(2N)$ |
| $2S(2n_0+n_1)/(2N)$ | $2S(n_1+2n_2)/(2N)$ |

Chi-square test for independence of rows and columns (null hypothesis):

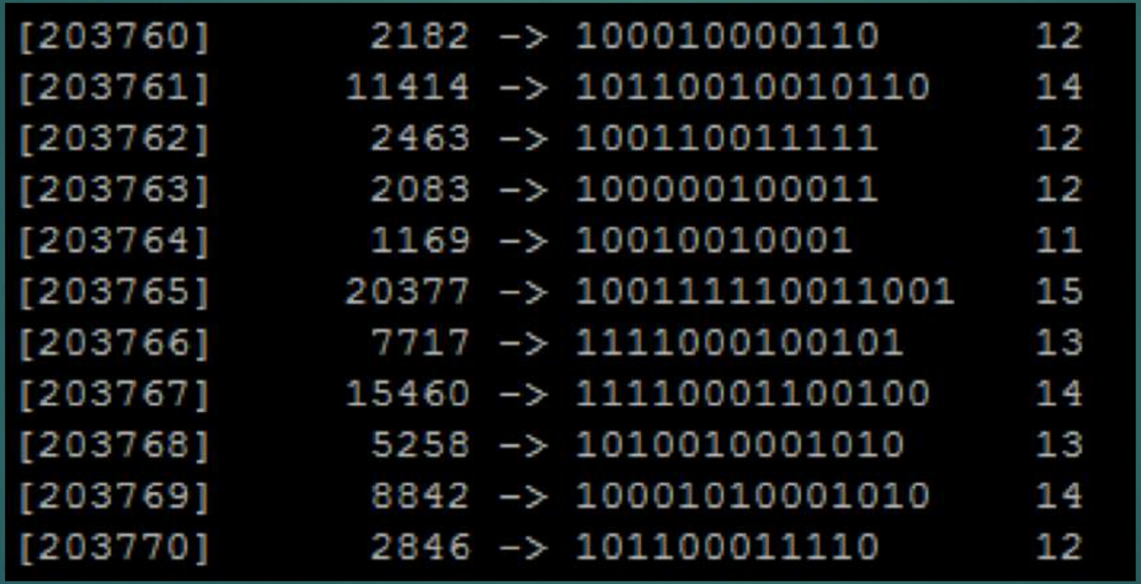
$$\sum \frac{(\text{Obs} - \text{Exp})^2}{\text{Exp}} \sim \chi^2 \text{ with 1 df}$$

# VCF (Variant Call Format)



```
##real id in 1000genome project: HG03518
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  TYPE
1       13380   rs571093408     C       G       100     PASS    heterozygous
1       15211   rs78601809      T       G       100     PASS    heterozygous
1       15820   rs2691315       G       T       100     PASS    heterozygous
1       18849   rs533090414     C       G       100     PASS    heterozygous
1       30923   rs806731        G       T       100     PASS    heterozygous
1       49298   rs200943160     T       C       100     PASS    heterozygous
1       52238   rs2691277       T       G       100     PASS    heterozygous
1       55164   rs3091274       C       A       100     PASS    heterozygous
1       62777   rs528401309     A       T       100     PASS    heterozygous
1       69897   rs200676709     T       C       100     PASS    heterozygous
1       82343   rs563238524     T       C       100     PASS    heterozygous
1       83084   rs181193408     T       A       100     PASS    heterozygous
1       86331   rs115209712     A       G       100     PASS    heterozygous
```

# Filtering & Variable Length Encoding



```
[203760]       2182 -> 100010000110       12
[203761]      11414 -> 10110010010110     14
[203762]       2463 -> 100110011111       12
[203763]       2083 -> 100000100011       12
[203764]       1169 -> 10010010001        11
[203765]      20377 -> 100111110011001    15
[203766]       7717 -> 1111000100101      13
[203767]      15460 -> 11110001100100     14
[203768]       5258 -> 1010010001010      13
[203769]       8842 -> 10001010001010     14
[203770]       2846 -> 101100011110       12
```

# Filtering VCF and Compressed Representation

- Only SNP ”ID” and ”TYPE” columns essential
- ”QUAL” and ”FILTER” can be removed during preprocessing
- ”CHROM”, ”POS”, ”REF”, ”ALT” can all be found via ”ID” from dbSNP

- Trim the ”rs” in front of ”ID”, represent as integer
- Sort by ”TYPE”, so that we don’t have to keep heterozygous/homozygous
- Keep a single integer to determine when ”TYPE” changes

# Variable Length Encoding

- Sort ”ID”s, grouped by ”TYPE”
- Keep only differences using the minimum number of bits needed
- Keep another small stream for bit-lengths, encoded by a Huffman Tree

- Example VCF Filtering/Compression: Actual VCF Size: 15,428,390 Bytes
- Heterozygous-Stream: 944,166 bits Homozygous-Stream: 428,921 bits Total Main Stream Size: 1,373,087 bits
- 5-bits/len Auxiliary Stream Size: 1,631,105 bits Huffman Encoded Auxiliary Stream Size: 1,070,458 bits
- Main Stream + Huffman Auxiliary Stream: 305,444 Bytes

# Preliminary Results

- 1000 case / 1000 control. ~300K-350K SNPs per VCF, ~5.5M unique SNPs
- SGX Enclave creation: 0.193381 seconds
- Remote Attestation: 0.002464 seconds
- Main Application: 49.990706 seconds
- SGX Enclave destruction: 0.034888 seconds

# Acknowledgements

- Can Kockan (Indiana University)
- Natnatee Dokmai (Indiana University)
- Oguzhan Kulekci (Istanbul Technical University)
- Steve Myers (Indiana University)

- The Cancer Genome Collaboratory
- Indiana University Precision Health Initiative