Sky Faber
University of California: Irvine

Luca Ferretti
University of Modena and Reggio Emilia

Challenge 1 – Task 1 and Challenge 2 – Task 2

Outline

- Challenge 1 Task 1
  - Overview
  - Encoding
  - Aggregation
  - Tuning

- Challenge 2 Task 2
  - Building Blocks
  - Input parsing
  - Edit Distance from PSI-CA
  - Optimizations + Performance
  - Hamming Distance from PSI-CA

Overview

Computing statistics using only sum homomorphic encryption scheme (Paillier)

Client (with pk):
- Encoding [and pre-computation] of the local data
- Encryption using sum homomorphic scheme

Server:
- Sum over encrypted aggregated data

Client (with sk):
- Decryption and disaggregation
- Computation of MAF and chi-square over plaintext

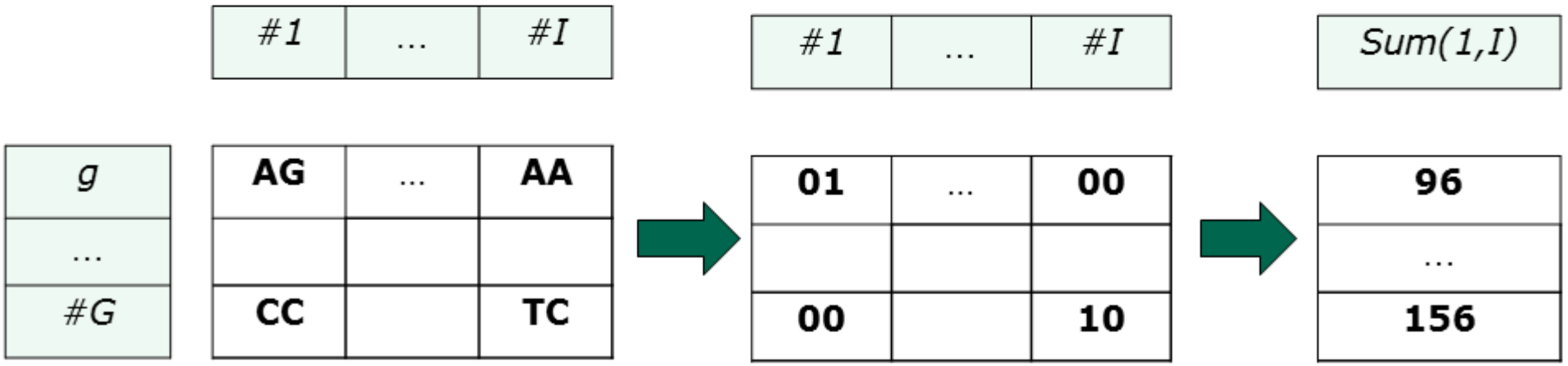
Client Encoding and [Pre-computation]

Sequence **encoding**: *Alleles sequence → 0/1 sequence*

Take a convention:

e.g. the *higher allele in increasing order is 1, the other is 0*

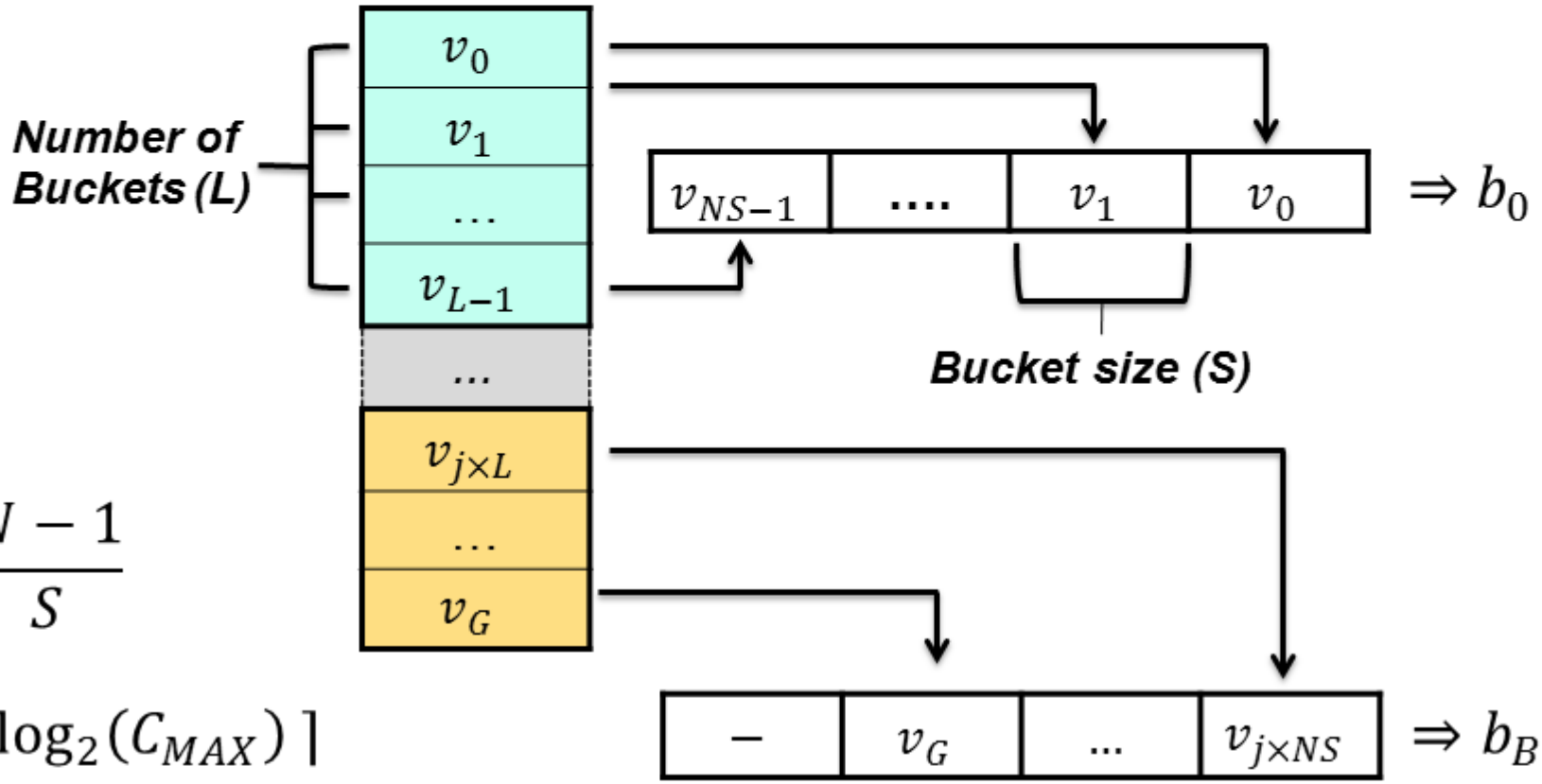
[Depending on the scenario, you may want to reduce each sequence to the sum of the encoded alleles before encryption.]



Client Aggregation

Aggregate many integers to a single big integer to reduce storage overhead and addition operations on the encrypted data.

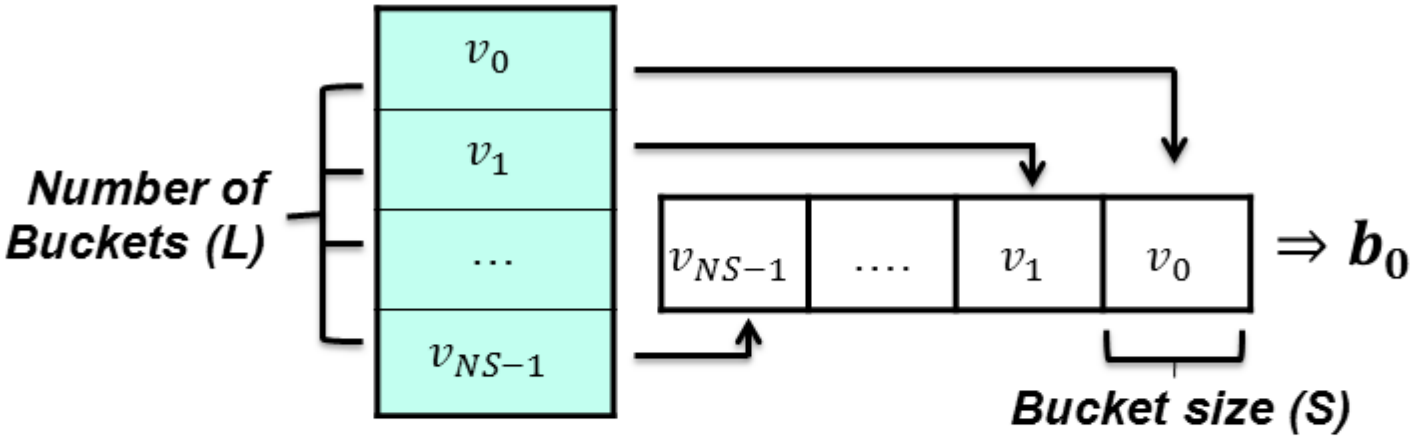
Each bucket should be sized to not overflow the total amount of samples (C_{MAX})



$$L = \frac{N-1}{S}$$

$$S = \lceil \log_2(C_{MAX}) \rceil$$

Client Aggregation – Tuning (1)



$$L = \frac{N-1}{\lceil \log_2(C_{MAX}) \rceil}$$

Even if increasing the ciphertext size (N) allows to store more buckets, performance decrease significantly. Thus it should be only a choice ***due to security constraints.***

Decreasing the maximum amount of summed samples greatly increase performance, but more information may leak about the original datasets.

Client Aggregation – Tuning (2),

Brief analyses of different sizing of C_{MAX} $(300\ gt \times 100\ ind)$ for N=1024 and N=2048*:*

$$C_{MAX} = 10^4 \quad \Rightarrow \quad B = 76,254 \quad \Rightarrow \quad Enc \sim 1.2\,, \quad 4.4s$$

$$C_{MAX} = 10^6 \quad \Rightarrow \quad B = 51,102 \quad \Rightarrow \quad Enc \sim 1.7\,, \quad 5.9s$$

$$C_{MAX} = 10^8 \quad \Rightarrow \quad B = 36,\ 76 \quad \Rightarrow \quad Enc \sim 2.4\,, \quad 7.7s$$

Sizing the maximum counter to a lower value than the total amount of samples decreases bandwidth overheads and of sum computed by the server, but:

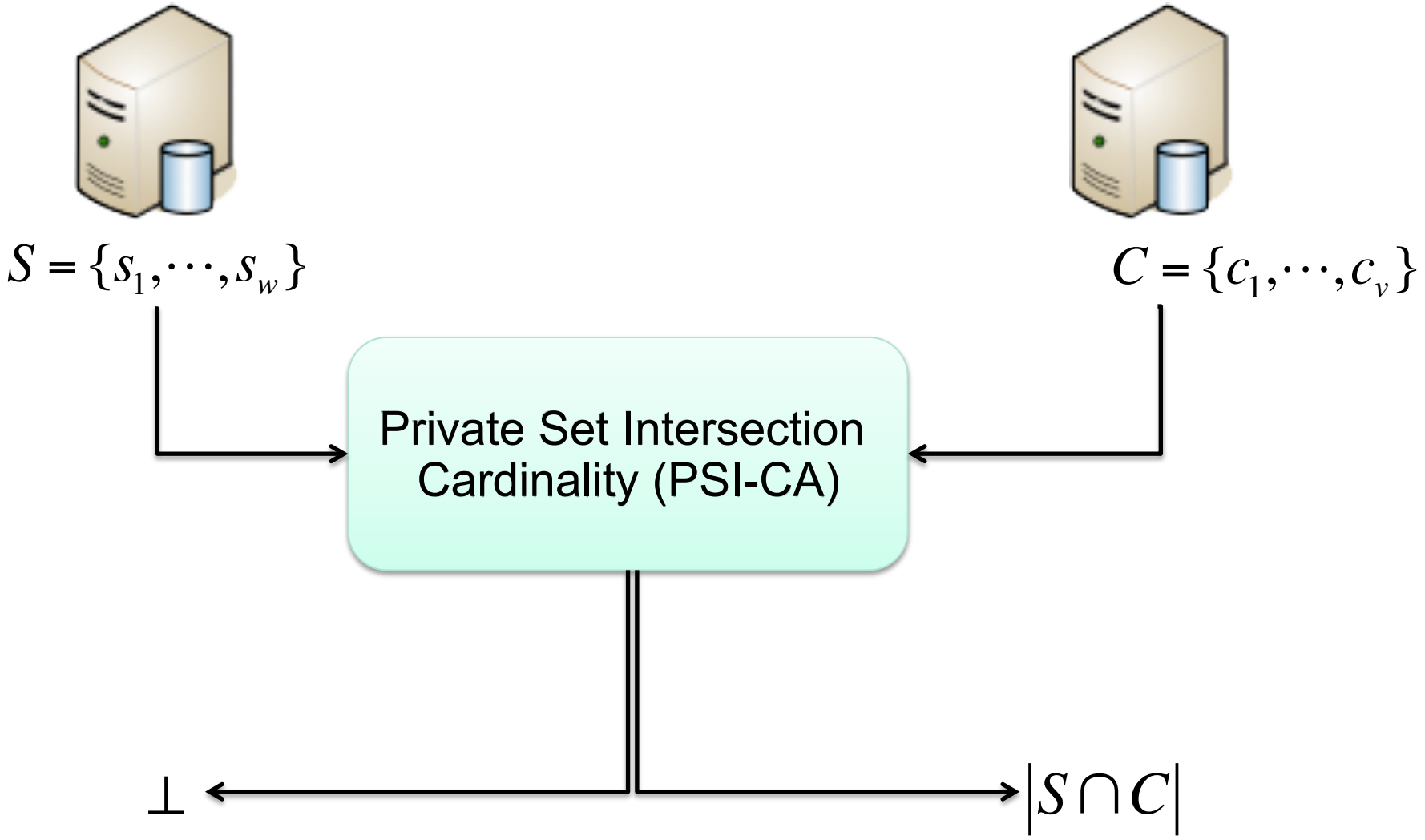
- the server should accumulate blocks depending on the amount of samples stored in each one
- the client with the decryption key may infer more information on the original partial datasets of the local organizations

Outline

- Challenge 1 Task 1
  - Overview
  - Encoding
  - Aggregation
  - Tuning

- Challenge 2 Task 2
  - Building Blocks – PSI-CA
  - Input parsing
  - Edit Distance from PSI-CA
  - Optimizations + Performance
  - Hamming Distance from PSI-CA

Building Blocks - Private Set Intersection Cardinality



Building Blocks – PSI-CA

*Must support randomization w/ inverse

Public Parameters $G^*, H(\cdot), H'(\cdot)$

$S = \{s_1, \cdots, s_w\}$

$R_s \leftarrow ord(G)$

$C = \{c_1, \cdots, c_v\}$

$R_c \leftarrow ord(G)$

$\forall i : a_i = H(c_i)^{R_c}$

$\longleftarrow$

$\forall j : ts_j = H'(H(s_j)^{R_s})$

$\forall i : a'_i = a_{\Pi(i)}^{R_s}$

$\longrightarrow$

$\forall i : tc_k = H'(a'^{R_c^{-1}}_i)$

$\perp$

$\left|\{ts_1, \ldots, ts_w\} \cap \{tc_1, \ldots, tc_v\}\right| = \left|S \cap C\right|$

Introduced in “Fast and private computation of cardinality of set intersection and union.” by De Cristofaro, Gasti, and Tsudik 2012

Input Processing

Idea – Process each record in VCF into pair (position, nucleotide)

SNP/SUB – For the string $s_1 s_2 ... s_n$ at offset p

Output : $\{(s_1, p), (s_2, p+1) ..., (s_n, p+n-1)\}$

DEL – For a del of length n at offset p

Output : $\{(-, p), (-, p+1) ..., (-, p+n-1)\}$

INS – For the string $s_1 s_2 ... s_n$ inserted at offset p

Output : $\{(s_1, p.1), (s_2, p.2) ..., (s_n, p.n)\}$

Notice all operations map to unique pairs

Reducing Edit distance to PSI-CA

Main Idea - use PSI-CA to count the similarities between genomes by counting common pairs.

As input give all sets of (position,nucleotide) pairs. Count of matching pairs returned

PROBLEM! – How do we convert a count of common base pairs to a count of differences when positions may not match.

Solution – Run PSI-CA again on the positions only

E.G. : S = {(3.3,A)}, C = {3,G}, Edit Dist. = 2, CA = 0
: S = {(3,A)}, C = {3,G}, Edit Dist. = 1, CA = 0

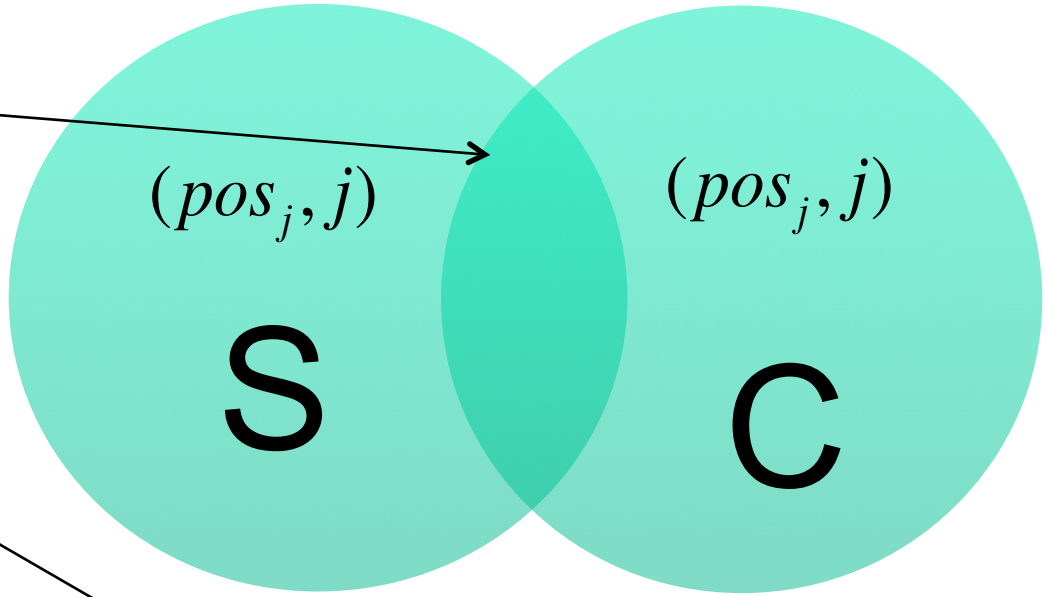
Reducing Edit distance to PSI-CA

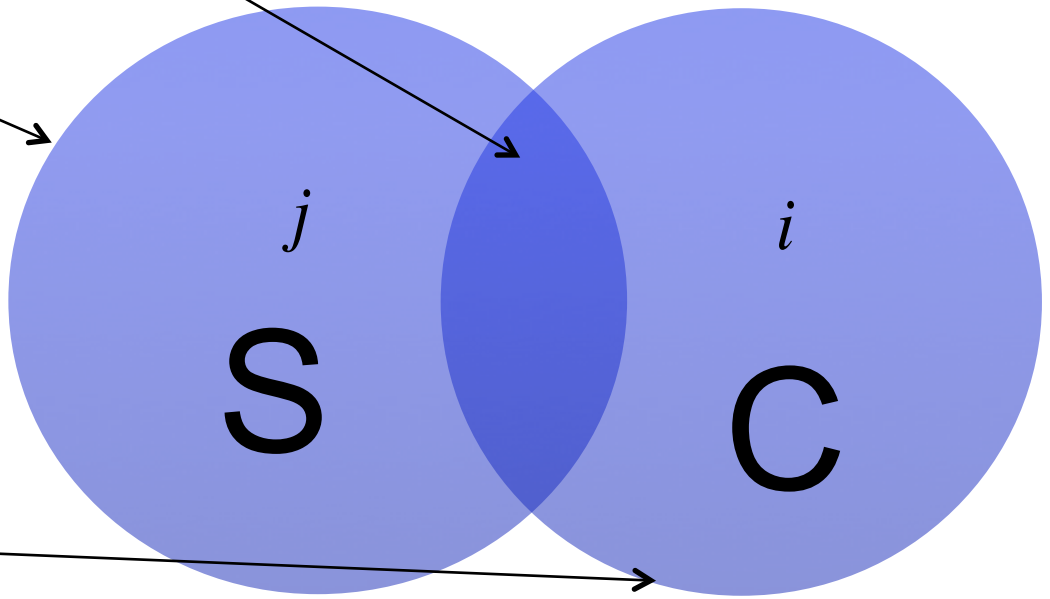
CB = Number of places where $pos_i = pos_j \wedge i = j$

CP = Number of places where $i = j$

w = size of S

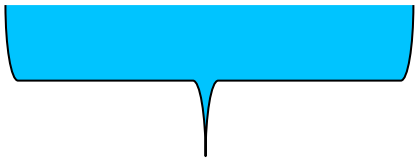
v = size of C





Reducing Edit distance to PSI-CA

Edit Distance = v + w – CP - CB



Number of unique positions between C and S

Still has some inaccuracies – only an upper bound

- Two multi nucleotide insertions at the same reference position, but shifted will count improperly
- Similar with rare, large substitutions

E.G: AGCG vs GCG will be calculated as 4

Optimizations + Performance

Pipelining – Process and send as soon as possible.

Threading – Run each instance of PSI-CA in parallel

Group Selection –

- EC group – Small bandwidth, slow randomization
- DH group – Larger bandwidth, blazing fast randomization
    - In the right group can have ~160 bit exponents

Protocol sends ~v+w group elements and v hashes
computes ~2v+w randomizations and v inverses

Introduced in “Genodroid: are privacy-preserving genomic tests ready for prime time?” by De Cristofaro, Faber, Gasti, and Tsudik 2012

Optimizations + Performance

Two patients VCFs -100k lines

run in <15 min
~30mb data transfered

About 20% increase in encryptions

Supporting Hamming Distance

Hamming Distance supported easily by modifying the input processing.

- Basic Hamming Distance (Best Performance)
  - Skip all INS and DEL
  - Don’t separate SUB into individual pairs
- Higher Accuracy Hamming Distance
  - Skip all INS and DEL
  - Separate SUB into individual pairs
- Highest Accuracy Hamming Distance
  - Skip all DEL
  - Separate SUB into individual pairs
  - Run the protocol once for SNP/SUB and once for INS
    - Final computation for INS modified slightly
  - 4 instances of PSI-CA, but same complexity

Security Discussion

- Security in the Random Oracle Model
- Secure only against Honest But Curios Adversaries
- Security against malicious adversaries could exist, but would be significantly slower. Would have to work around H'()