Private Set Intersection

Mădălina Bolboceanu

Bitdefender
theoretical research

Workshop on Selected Topics in Cryptography
March 23, 2022

What is this talk about?

- Our Python implementation of a PSI protocol

https://github.com/bit-ml/Private-Set-Intersection.

- This protocol was published by a team from Microsoft Research and academia

https://eprint.iacr.org/2017/299.pdf

https://eprint.iacr.org/2018/787.pdf

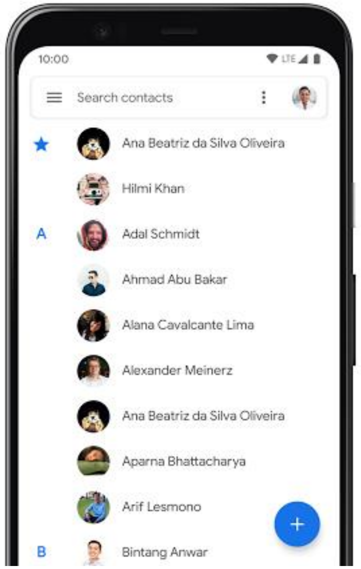
Outline

- 1. Motivation
- 2. Private Set Intersection (PSI)
- 3. Homomorphic Encryption (HE)
- 4. How to use HE to get PSI

- 5. The PSI protocol
- 6. Our Python implementation
- 7. Concluding remarks

1. Motivation

Question: who has WhatsApp installed in my contact list?



list of Client's contacts →

← contacts who have WhatsApp installed

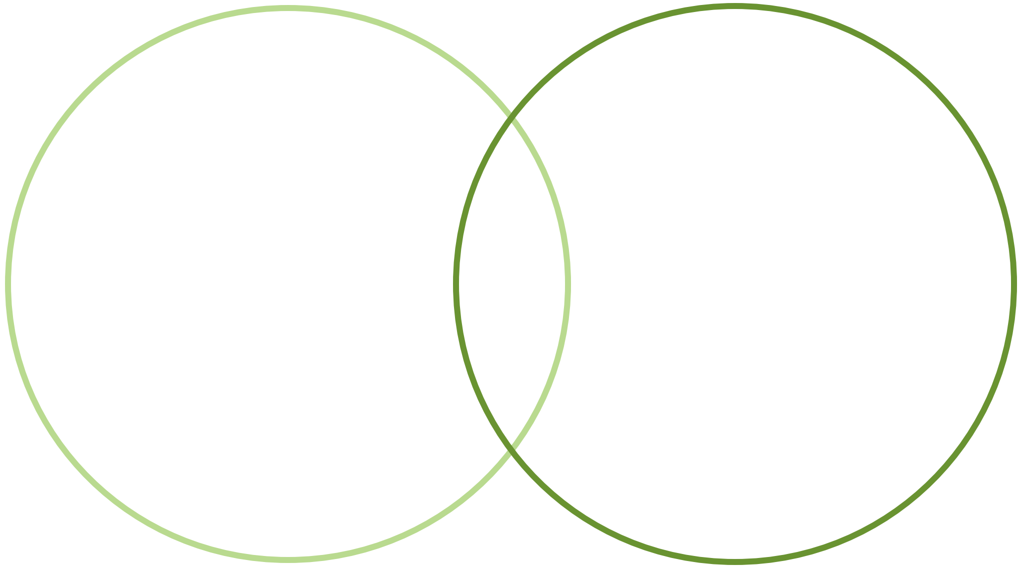
Client

WhatsApp Server

✓ The privacy of WhatsApp is protected.

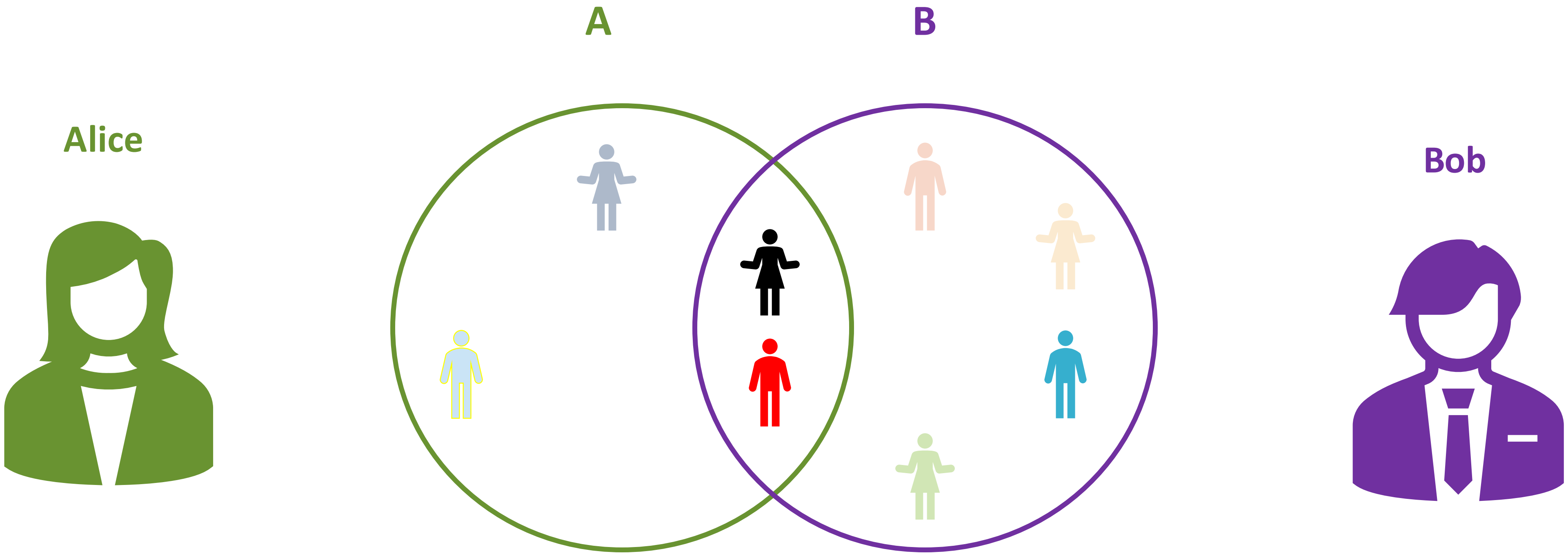
✗ WhatsApp learns Client's list of contacts.

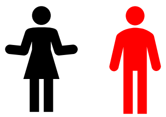
2. Private Set Intersection (PSI)



What is PSI?

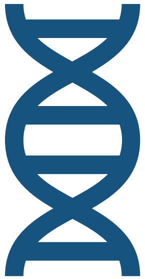
… An interactive crypto protocol.



Alice learns  and nothing else about set **B**.

Bob learns nothing about set **A**.

Other use cases of PSI



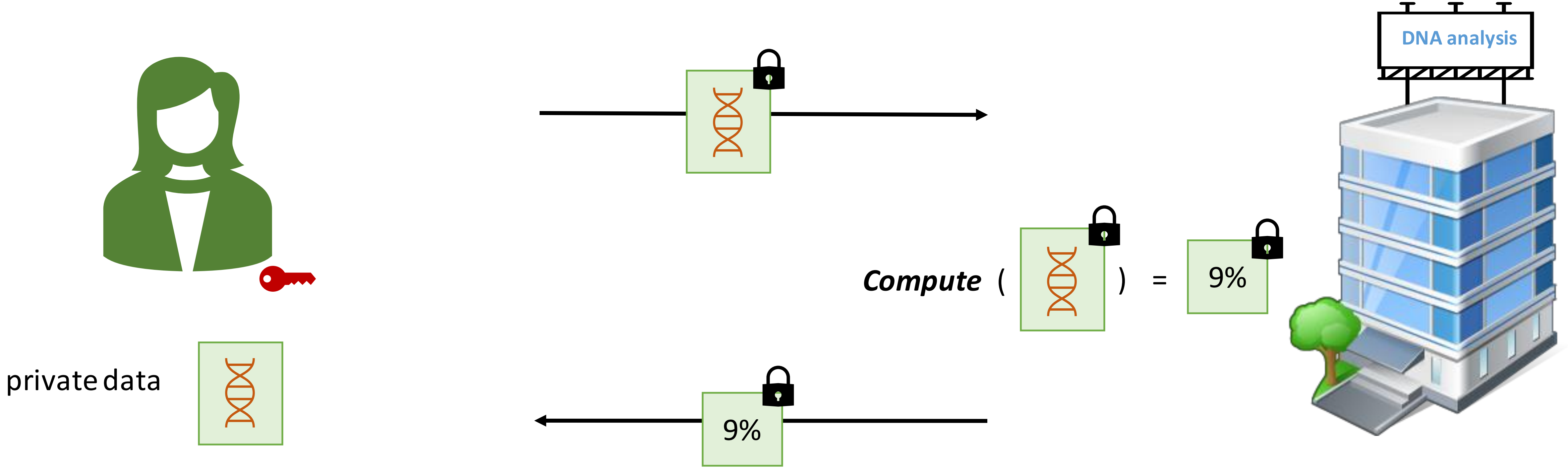
DNA private matching

Checking leaked passwords

Measuring ads efficiency

3. Homomorphic Encryption (HE)

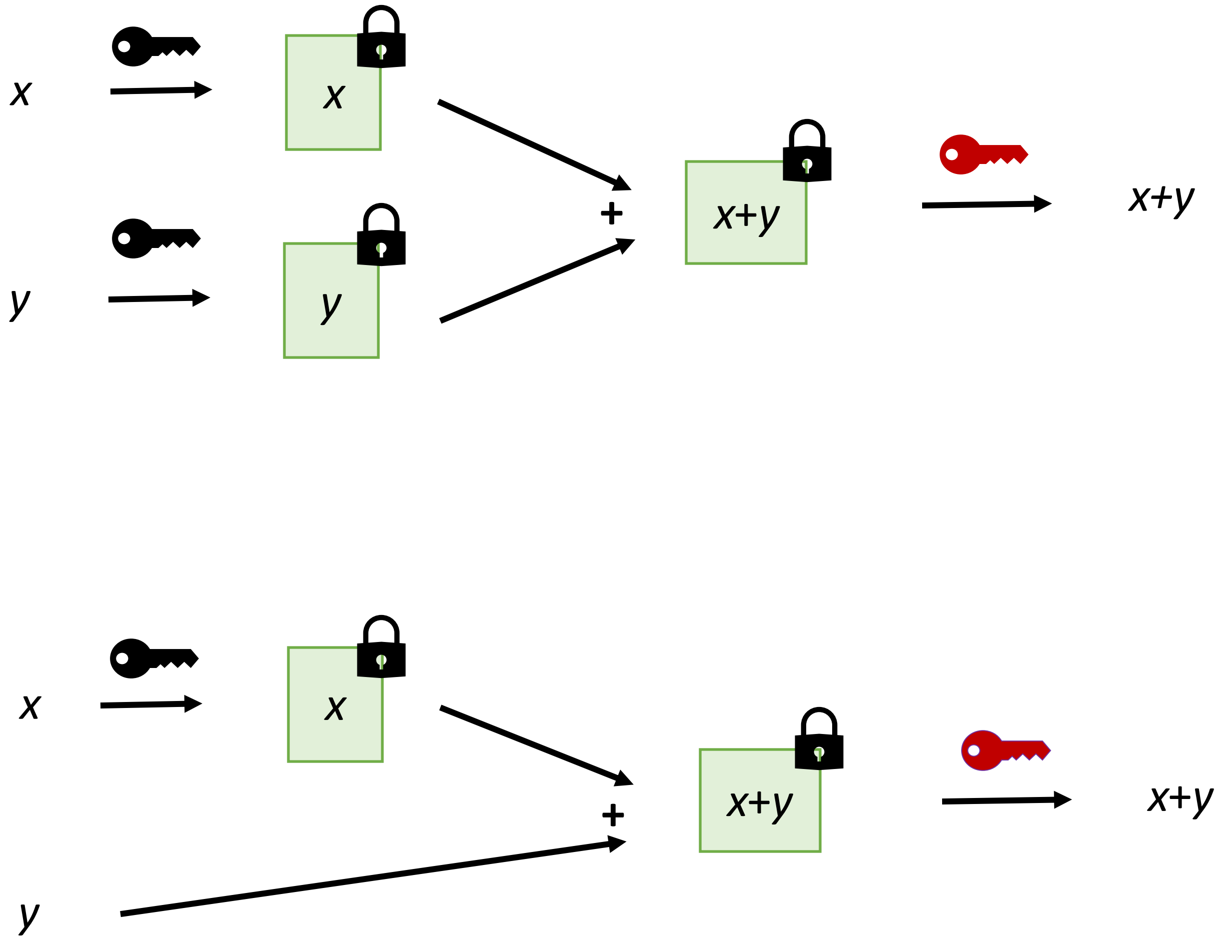
Homomorphic Encryption: an amazing tool

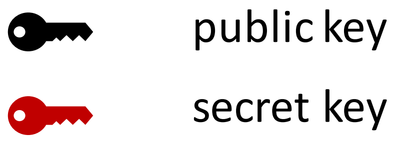


Our blogpost on HE: https://bit-ml.github.io/blog/post/homomorphic-encryption-toy-implementation-in-python/

What is the *Compute* function about?

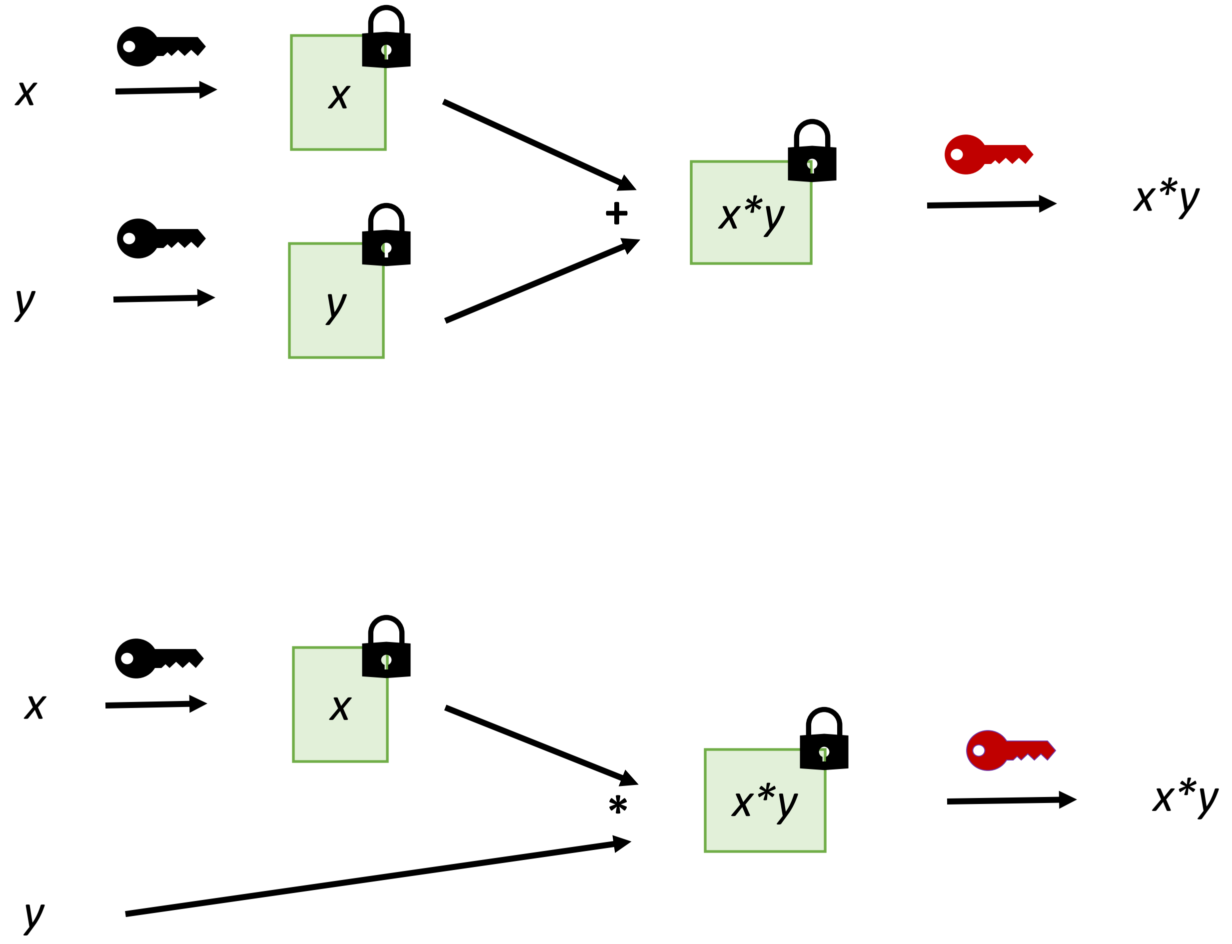
... Additions ...

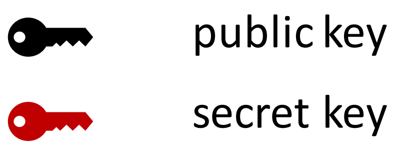




What is the *Compute* function about?

… and multiplications

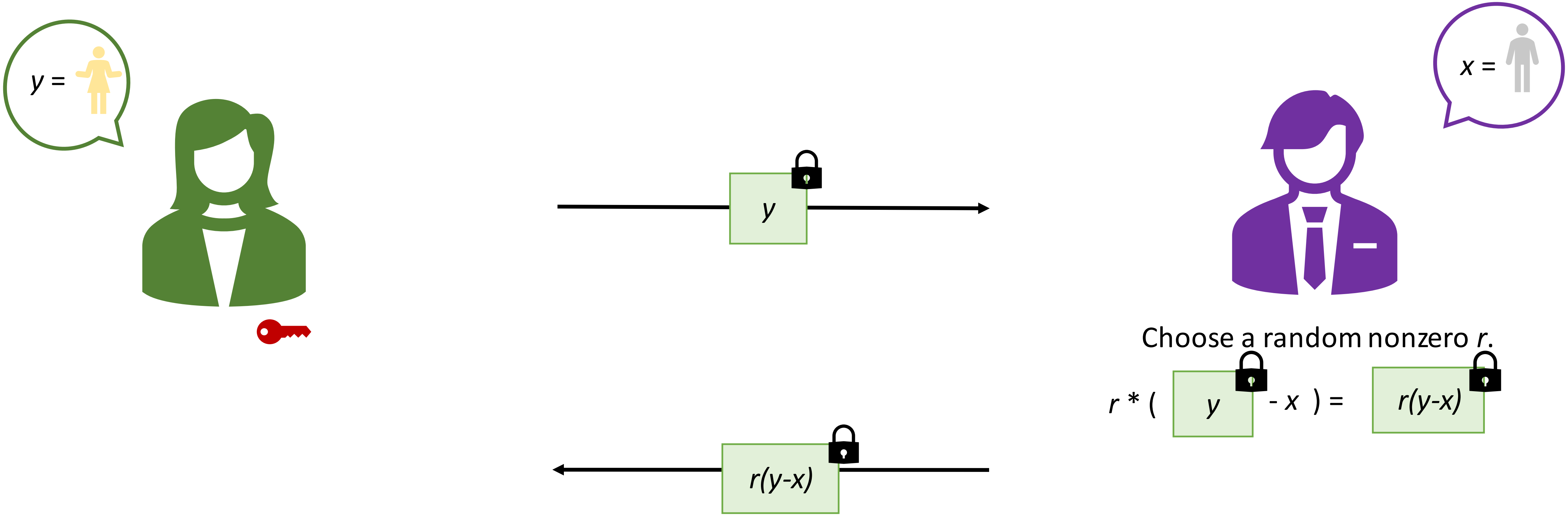




4. How to use HE to get PSI

A simple case: Just one friend...

x, *y*, *r* integers.



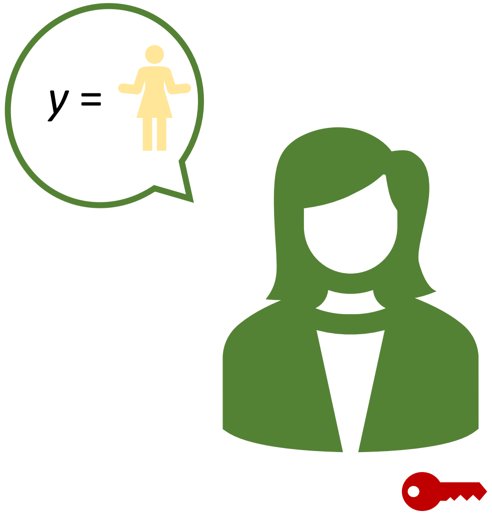
Decrypt and get *r(y-x)*.

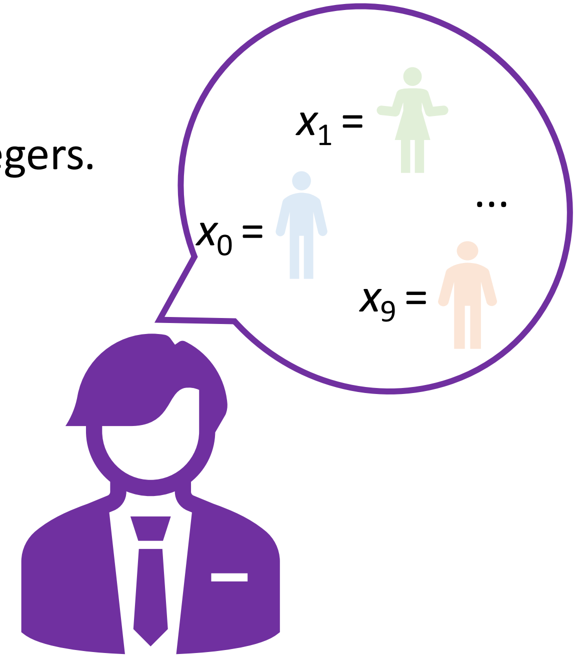
If it is 0, *y=x*.
Else, *y!=x.*

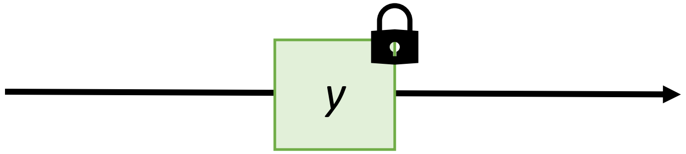
if *r* stays secret to Alice*, r(y-x)* will look random to Alice and hence won't get any information on *x.*

One friend, many friends...

$y, x_0, x_1, \ldots, x_9, r$ integers.







Choose a random nonzero r.
Compute

$$r * (y - x_0) * (y - x_1) * \ldots * (y - x_9)$$

$$= r(y-x_0)(y-x_1)\ldots(y-x_9)$$

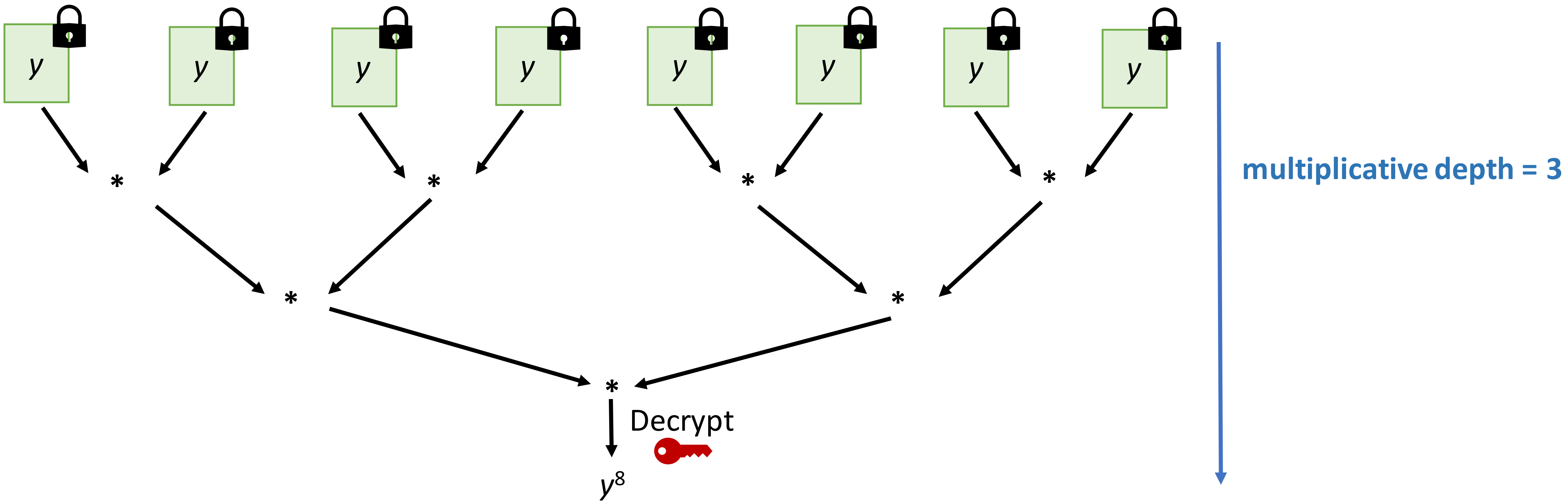
$r(y-x_0)(y-x_1)\ldots(y-x_9)$

Decrypt and get $r(y-x_0)(y-x_1)\ldots(y-x_9)$.

If it is 0, y is among Bob's friends. Else, y is not.

Warnings

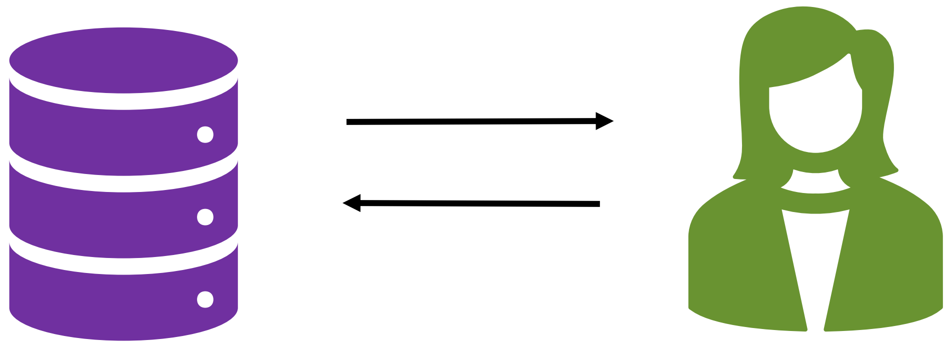
A typical HE scheme can't support too many multiplications.



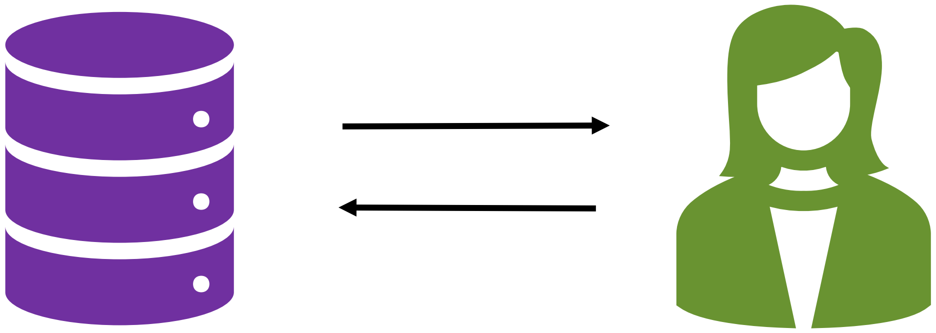
The communication increases linearly with the number of friends Alice has.

How to make the protocol more practical?

5. The PSI protocol



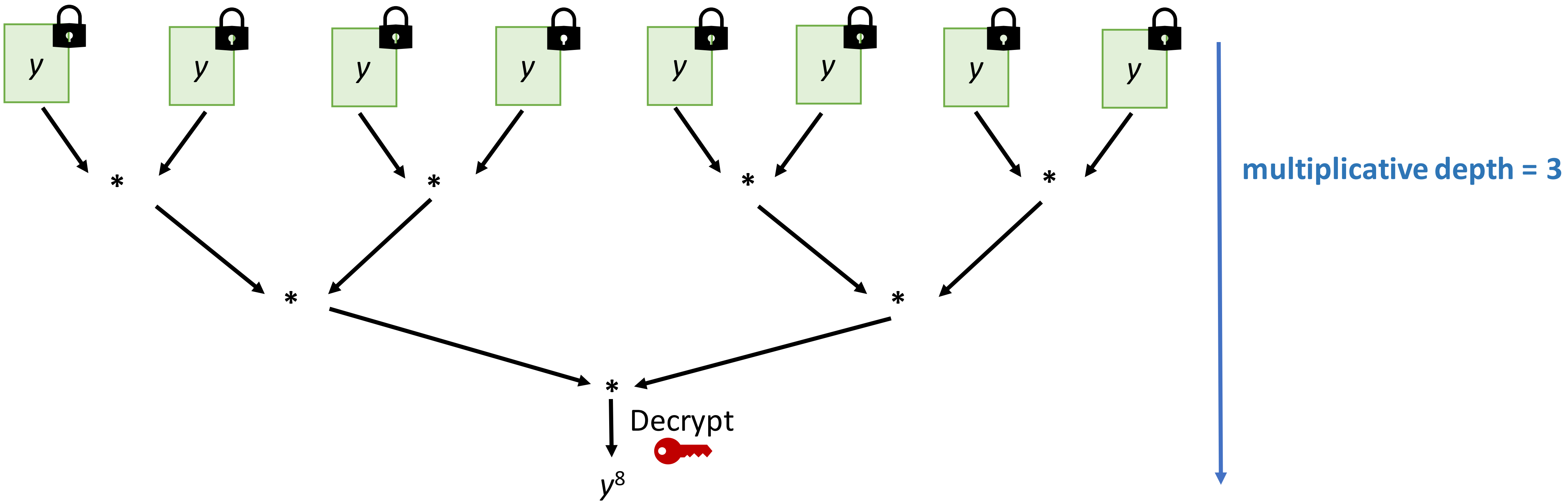
Meet the players of the PSI protocol



Server holds a set X.

Client holds a set Y.
Client wants to learn $X \cap Y$.

How large can the server's database be?



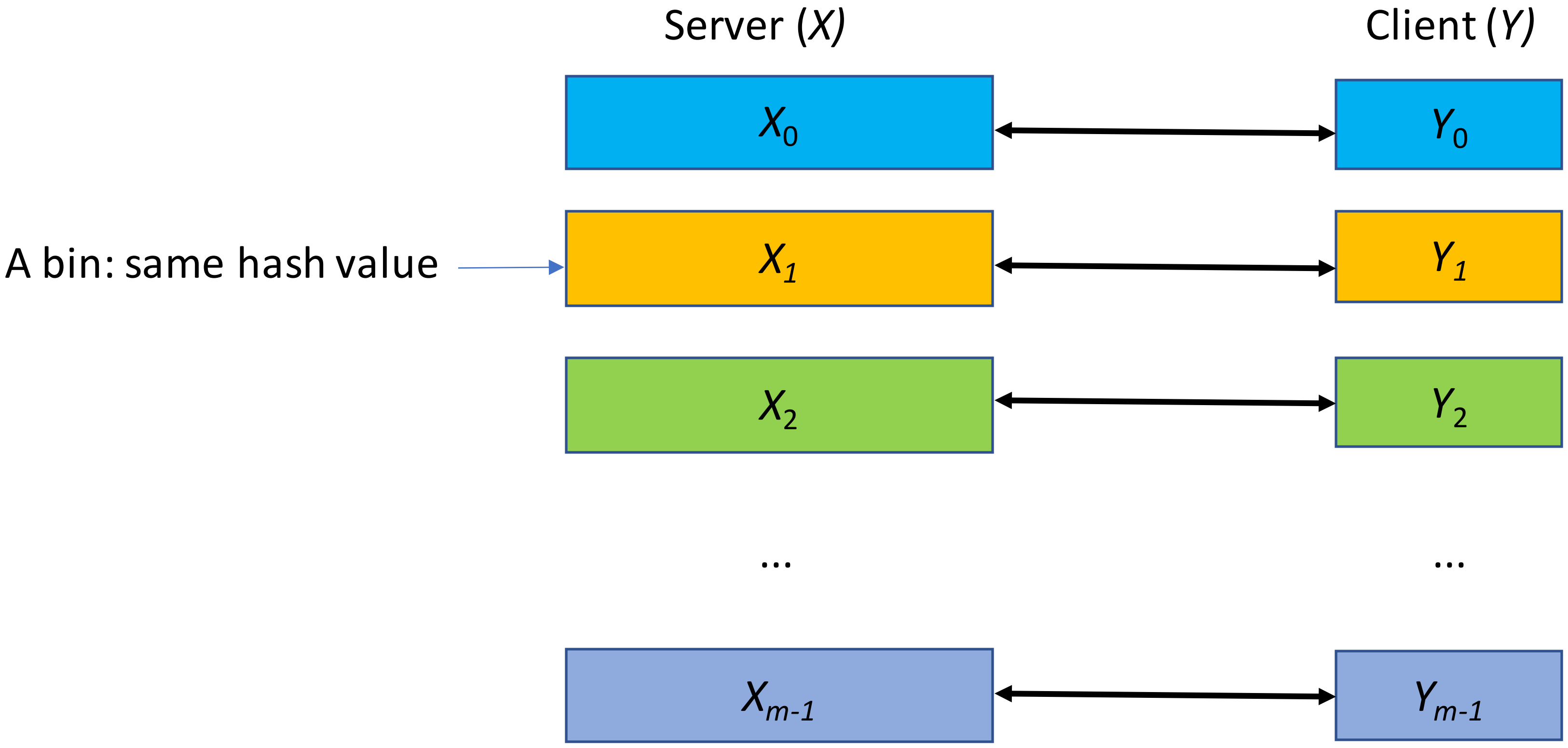
So, only 8 elements?

But, in general, the server has thousands of elements ...

How to deal with this?

1. Hashing

Suppose X, Y subsets of U. Partition X and Y using a hash function $h : U \rightarrow \{0,1,\ldots,m\text{-}1\}$.



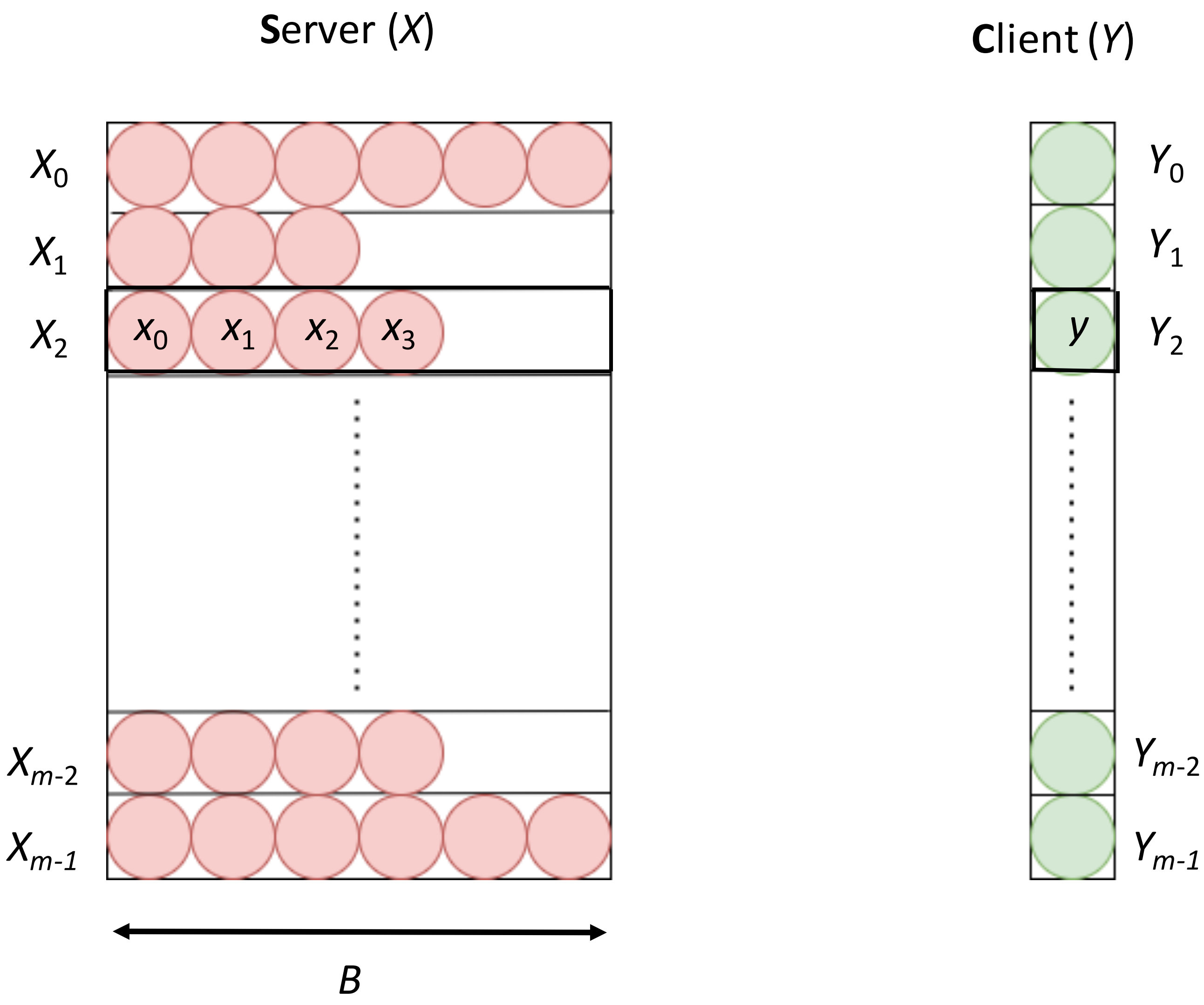
Now each X_i can have at most 8 elements.
Run the PSI protocol for each pair (X_i, Y_i).

About hashing

C applies Cuckoo hashing with 3 hash functions.
No collision if $m \sim 1.5 * |Y|$ w.h.p.

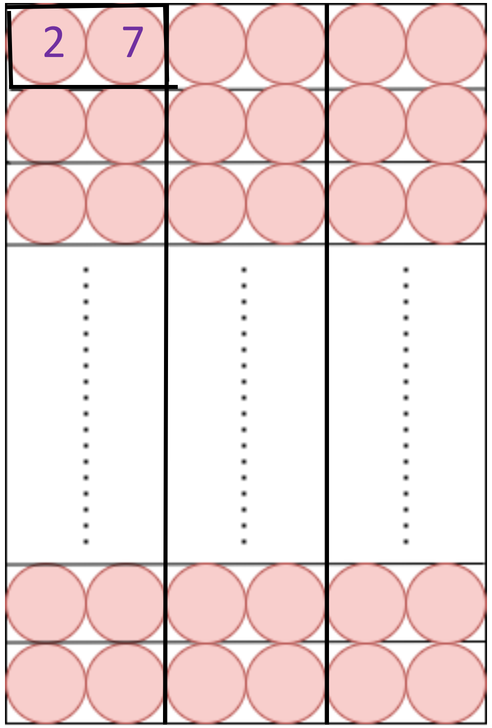
S applies simple hashing using same functions.
B is chosen such that the hashing almost never fails.

Both **C** and **S** apply padding.

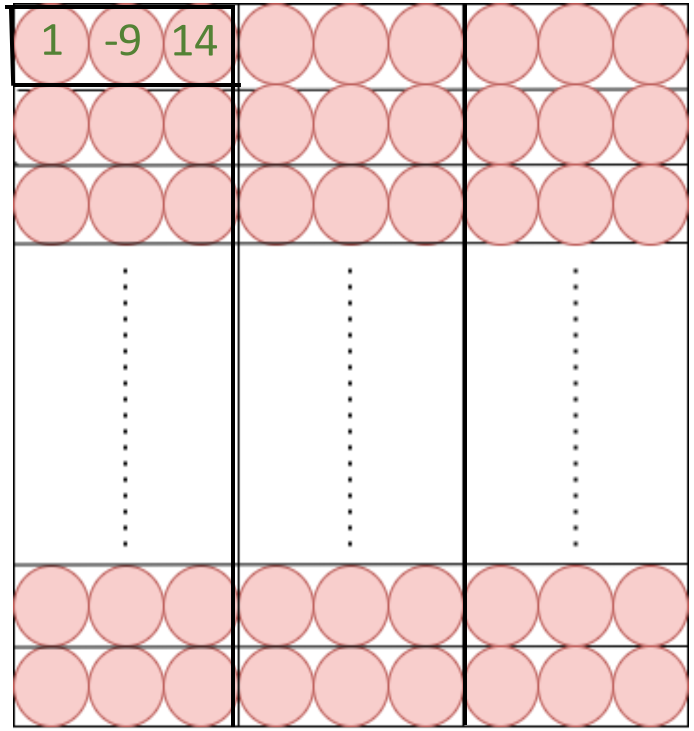


2. Partitioning and finding the polynomials

Server partitions each bin into α mini bins and associates to each of them a polynomial.

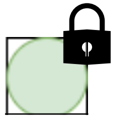


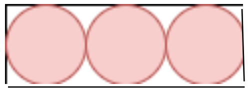
$(x-2) * (x-7) = 1*x^2 - 9*x + 14$

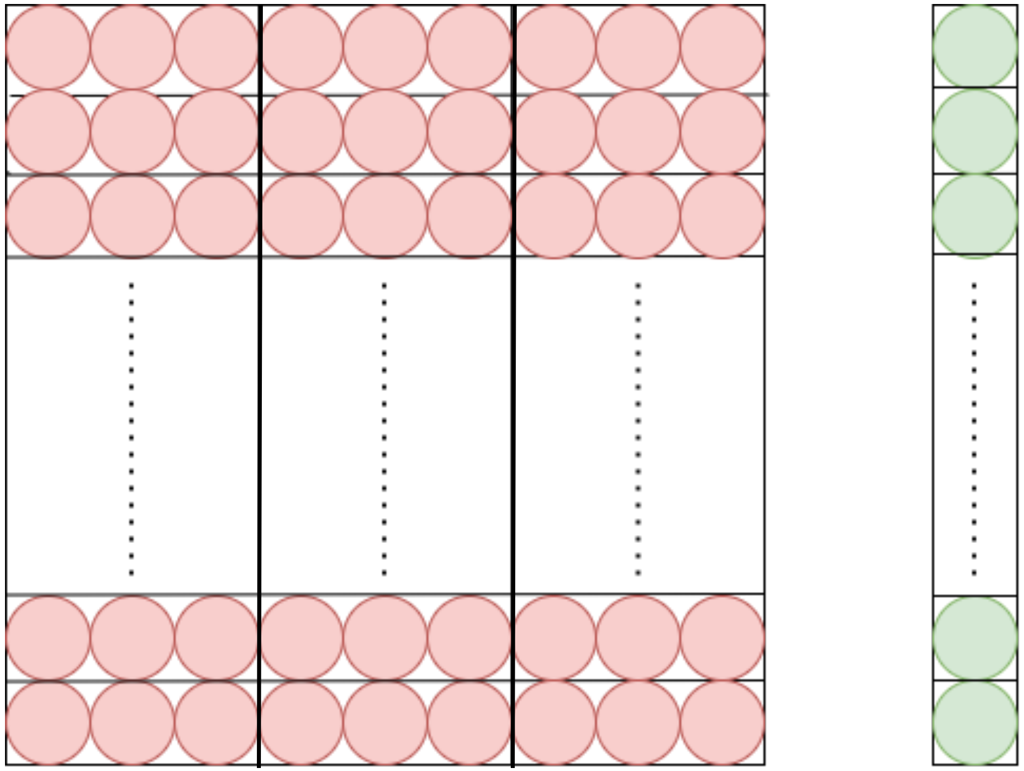


At this stage...

Server and Client can repeat the initial PSI protocol

for every 

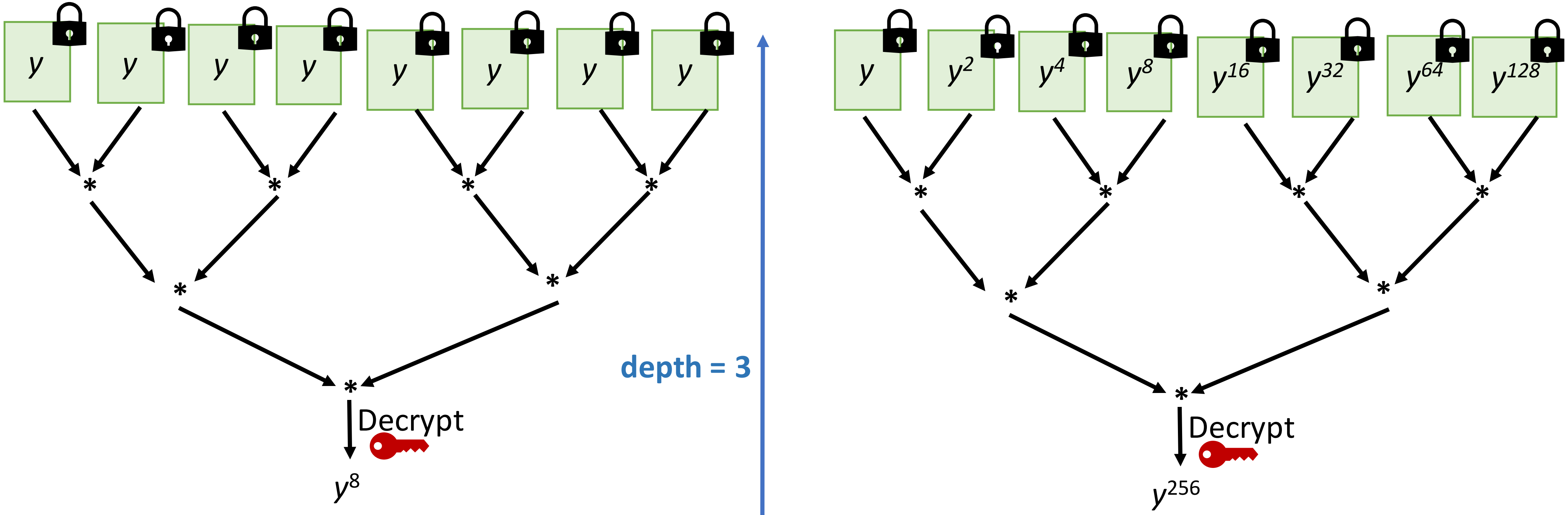
and every corresponding  .



We reduced the degrees of polynomials involved.

The Server to Client communication is increased by a factor of α.

3. Windowing: no need to reduce degrees that much



Client can send y y^2 … y^{128} instead of just y .

About windowing

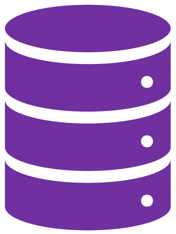
✓ Server can have a larger database.

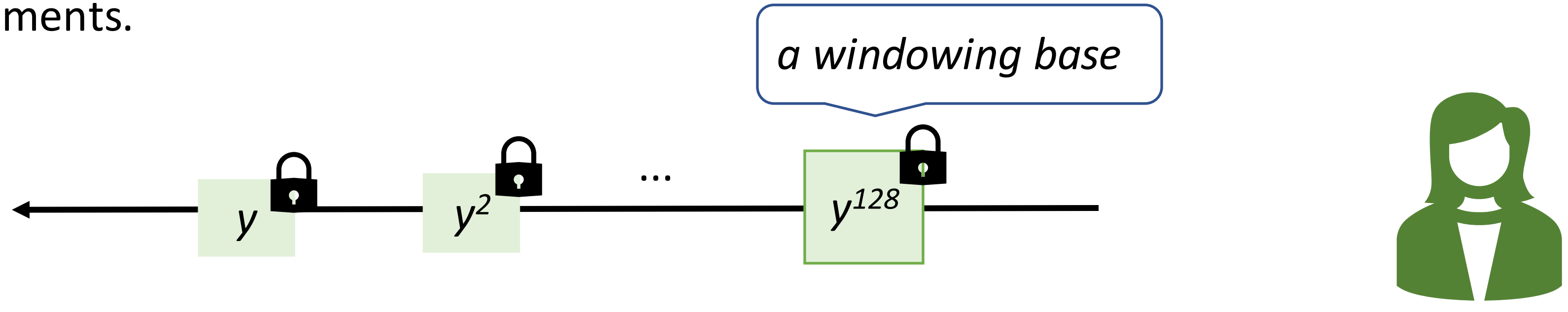
✗ It increases the Client to Server communication.

! Client can further lower the depth if he uses base 2^L instead of base 2.

How windowing is used

Suppose a mini bin has 255 elements.





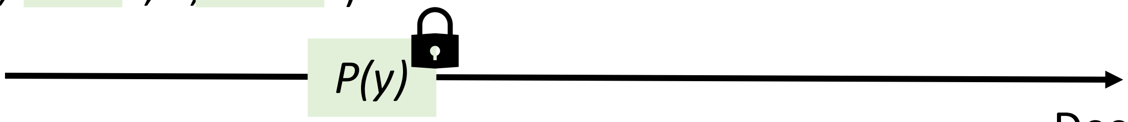
Take *P* the associated polynomial of the mini bin and its coeffs, $p_0, \ldots, p_{255}$.

For each k ≤ 255:

Write $k = k0 + k1 * 2 + \ldots + k7 * 2^7$.

Compute $y^k = (y)^{k0} * (y^2)^{k1} * \ldots * (y^{128})^{k7}$

Compute $P(y) = \langle (p_0, \ldots, p_{255}), (1, y, \ldots, y^{255}) \rangle$



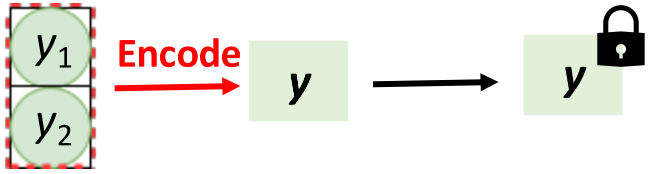
Decrypt.

If *P(y)* = 0, then *y* is in the mini bin.

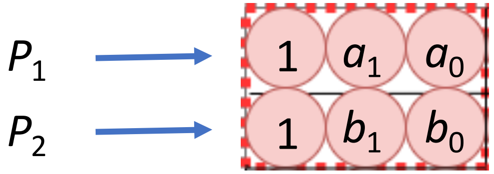
4. Batching

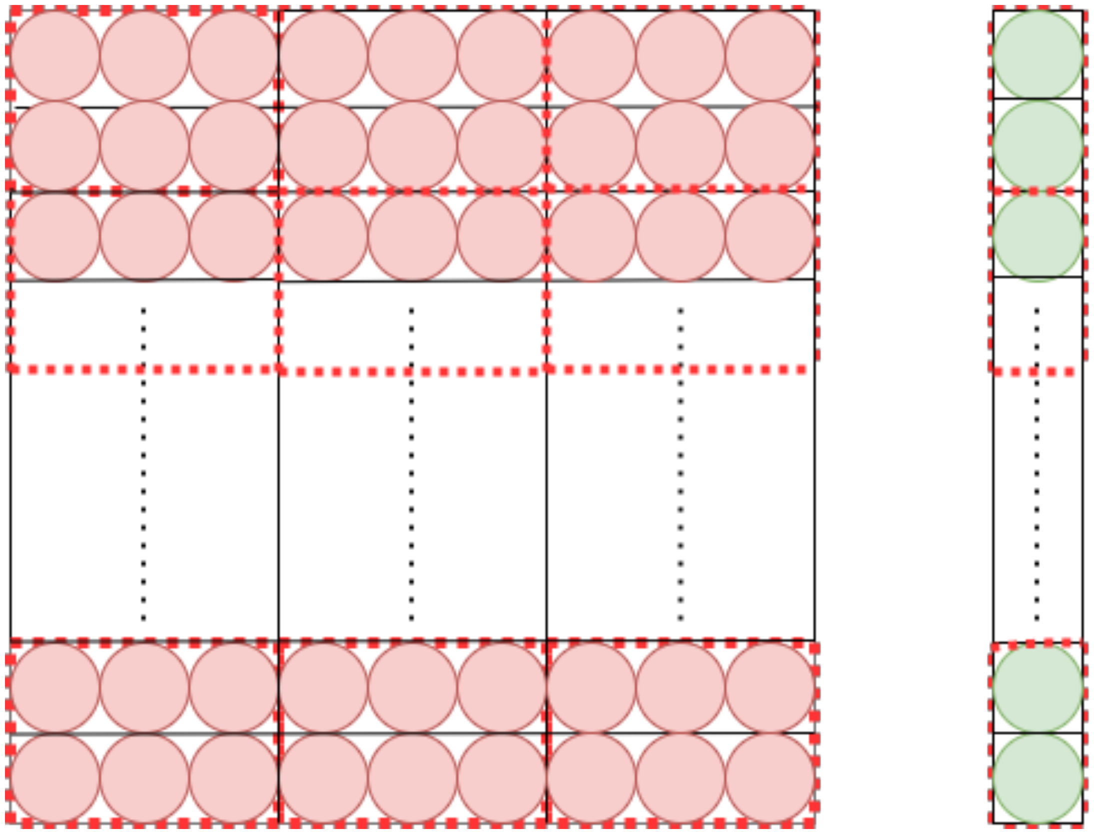
! CRT-like encoding

✓ Client "batches" many elements into one element to encrypt.



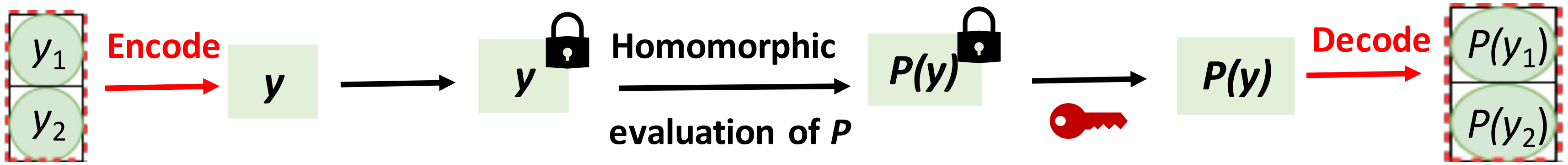
✓ Server "batches" the corresponding mini bins and evaluates their polynomials P_1 and P_2 at once.





About batching

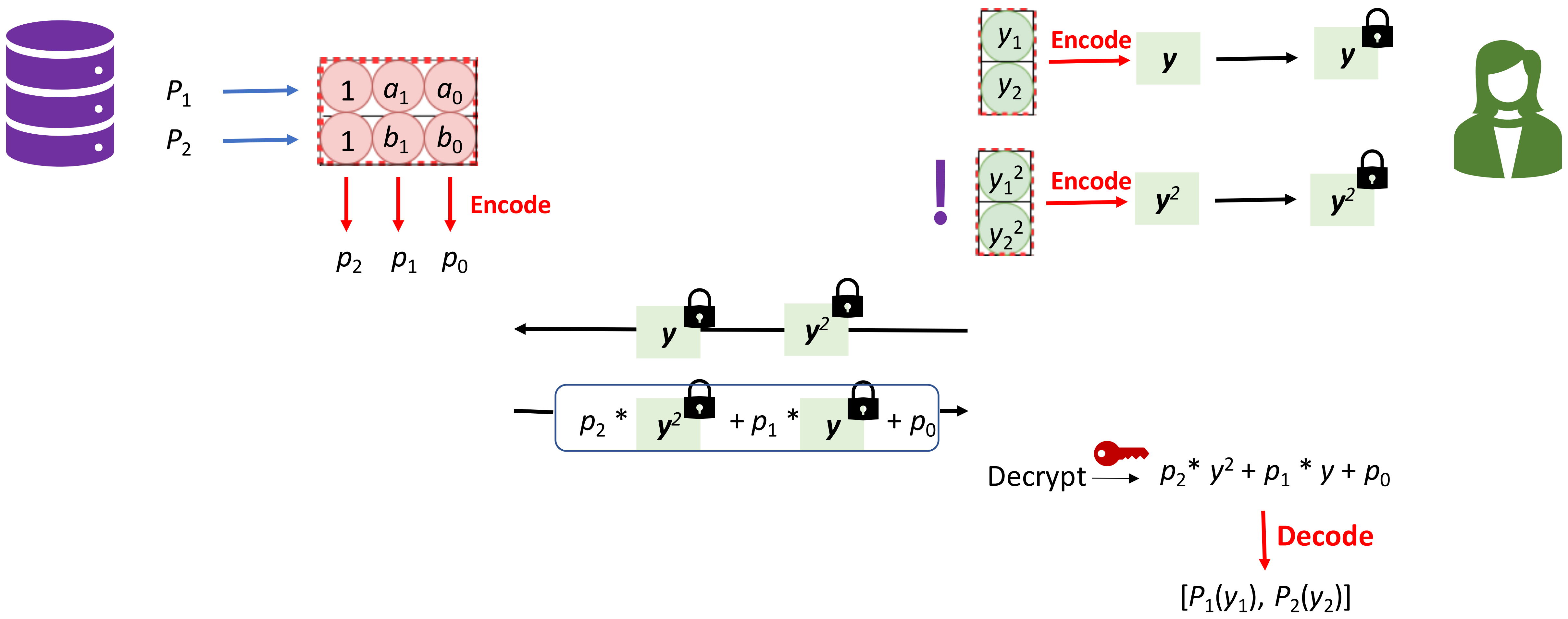
Server performs Single Input Multiple Data (SIMD) operations on ciphertexts.



✓ It reduces the Client to Server communication.

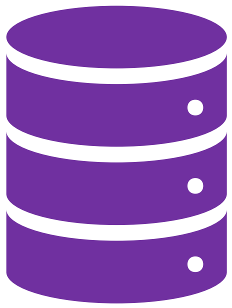
✓ It reduces the Server computation time.

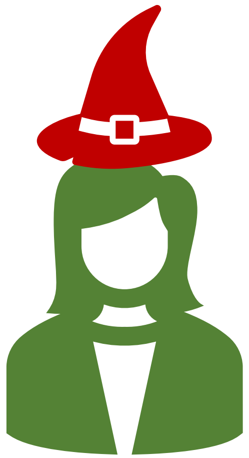
How batching is used



Client can check simultaneously if $P_1(y_1) = 0$ (i.e. y_1 is in the 1st mini bin) and $P_2(y_2) = 0$ (i.e. y_2 is in the 2nd mini bin).

Security



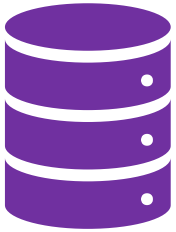


knows  and randomness used in the HE scheme.

✗ can learn info about the polynomials of the server.

✗ can learn info about server's set *X*.

Oblivious PRF assures privacy against malicious client!



Server: X, a secret key .

Computes X' = X

Client: Y.

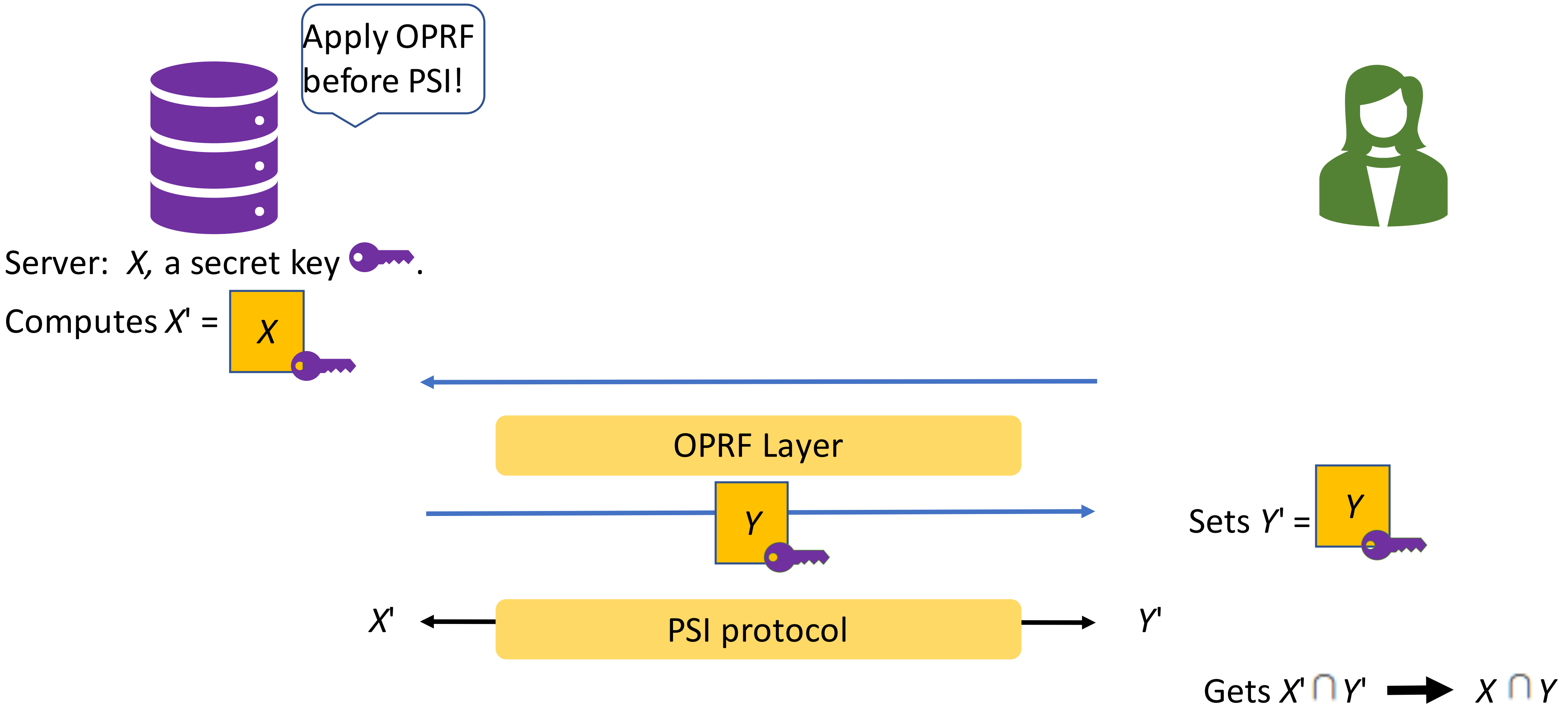
OPRF Layer

Y

Sets Y' = Y

learns nothing about X from X' unless she knows .

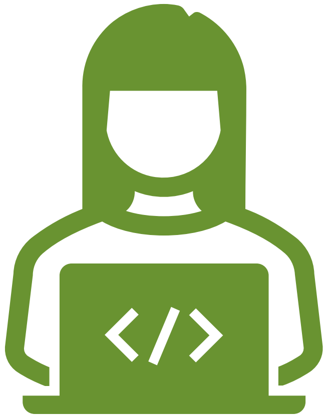
About OPRF



! It is a Diffie-Helmann-like protocol using elliptic curves point additions.

6. Our Python implementation

joint work with Miruna & Radu



Our implementation

- We use TenSEAL [1], a Python library for doing HE operations, built on top of Microsoft SEAL.

- We use Brakerski-Fan-Vercauteren12 homomorphic encryption scheme [2].

[1] A. Benaissa, B. Retiat, B. Cebere, A.E. Belfedhal, "TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption", ICLR 2021 Workshop on Distributed and Private Machine Learning, https://github.com/OpenMined/TenSEAL.

[2] J. Fan, F. Vercauteren, Somewhat Practical Fully Homomorphic Encryption, https://eprint.iacr.org/2012/144.pdf

Short recap

Server

- OPRF encoding
- Simple hashing
- Partitioning
- Finding the polynomials
- Batching

- Polynomial evaluations

OPRF interaction

Send query

Send answer

Client

- OPRF encoding

- OPRF computations
- Cuckoo hashing
- Batching
- Windowing

- Find the intersection

offline

online

Time and communication size for |C| = 5000, |S| = 1 mil.

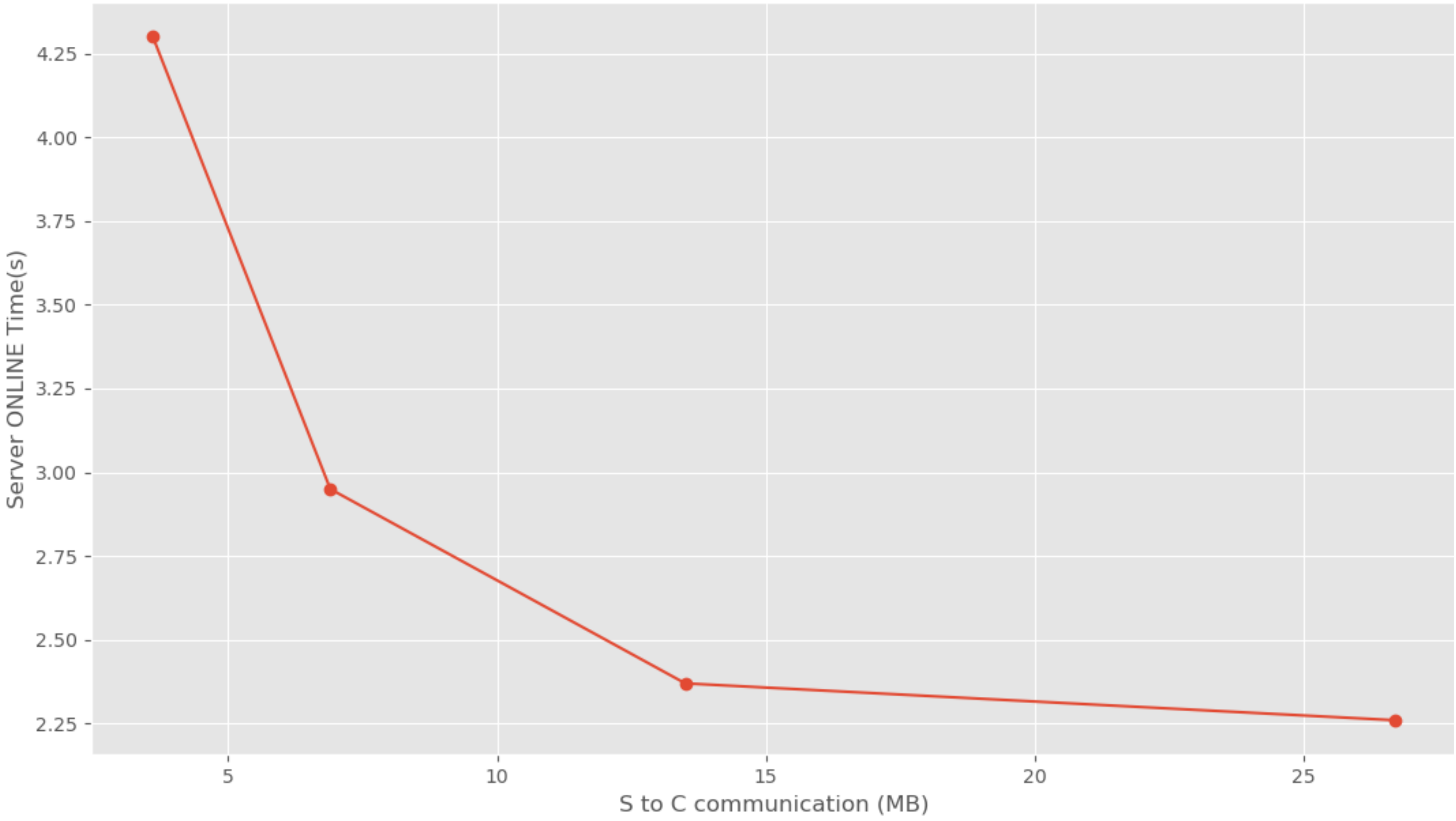
Time	Client (C)	Server (S)
online	1 s	3 s
offline	1 s	90 s

Communication **C->S: 5 MB**

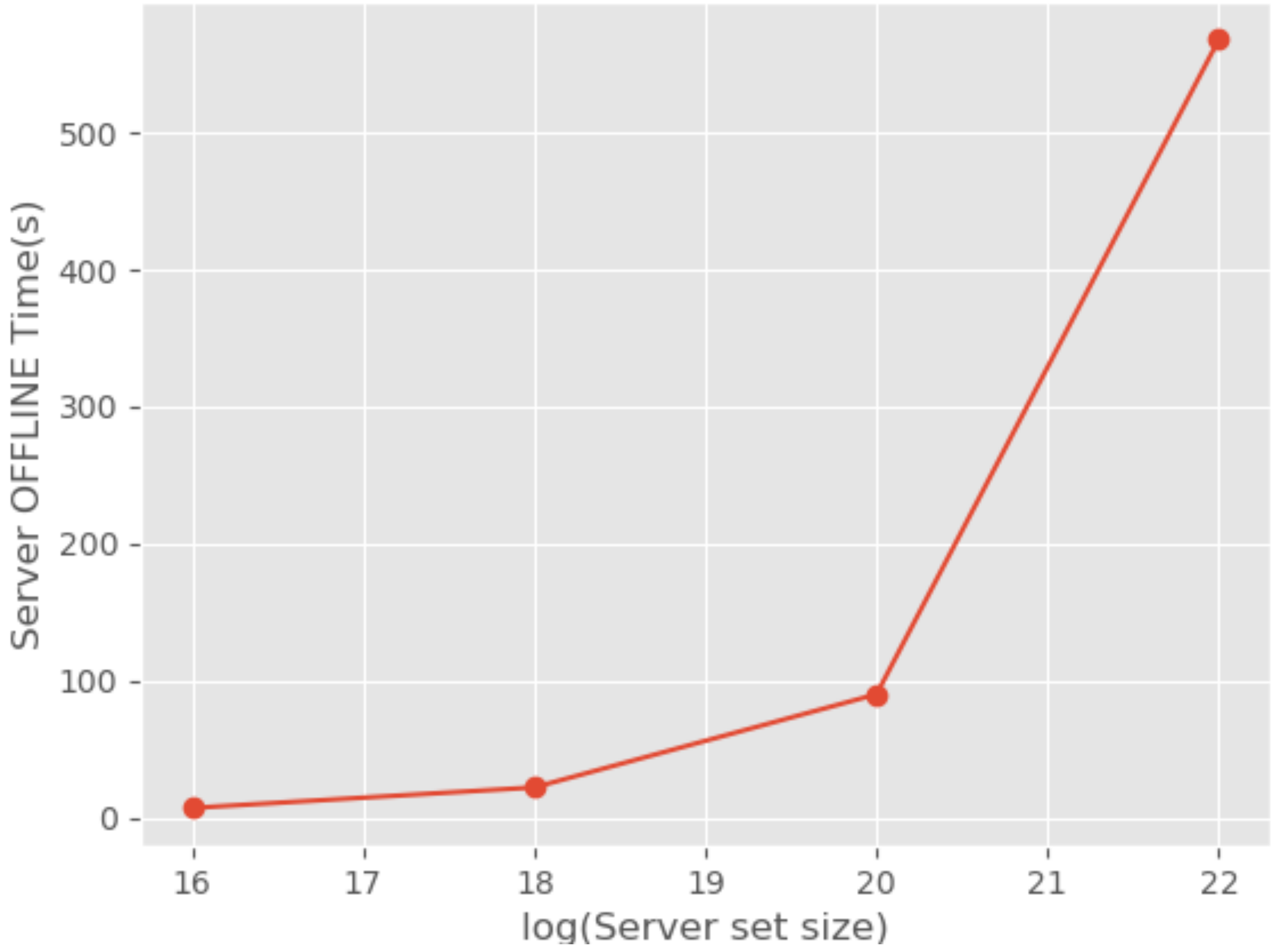
S->C : 7 MB

! Offline/Online time for the Client is always "small".

Server size 1 mil., time/communication trade-off



Server offline time



7. Concluding remarks

Takeaway

- Skipped many details of the protocol/ implementation.

Blogpost: https://bit-ml.github.io/blog/post/private-set-intersection-an-implementation-in-python/.

Code: https://github.com/bit-ml/Private-Set-Intersection.

- Many computations can be further parallelized.
- Considerable speed-up if you write it in C or C++.

PSI is a cool primitive with many interesting real world applications.

Many recent optimizations

!

Microsoft Research & academia published a new paper:

https://eprint.iacr.org/2021/1116.pdf.

They proposed and implemented in C++ several improvements of the previous protocol:

https://github.com/microsoft/APSI/.

Optimizations include:

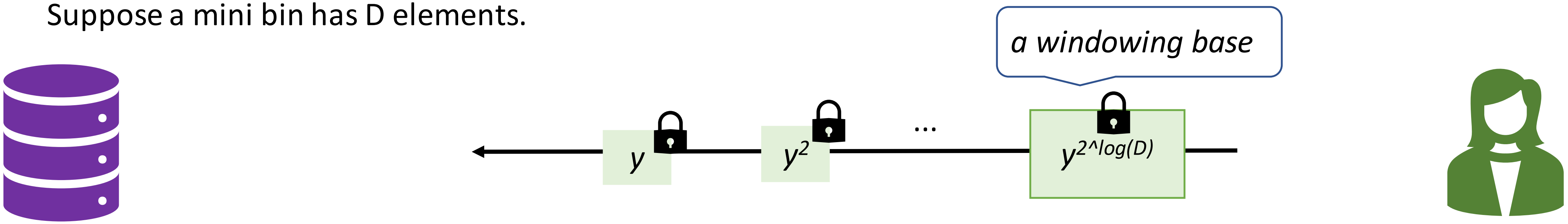
- a novel way of computing polynomial evaluations;
- a new *windowing-equivalent* procedure, etc.

✓ The Server computation time is improved.

✓ The Client to Server communication is reduced.

Improving windowing: using global postage bases

Suppose a mini bin has D elements.

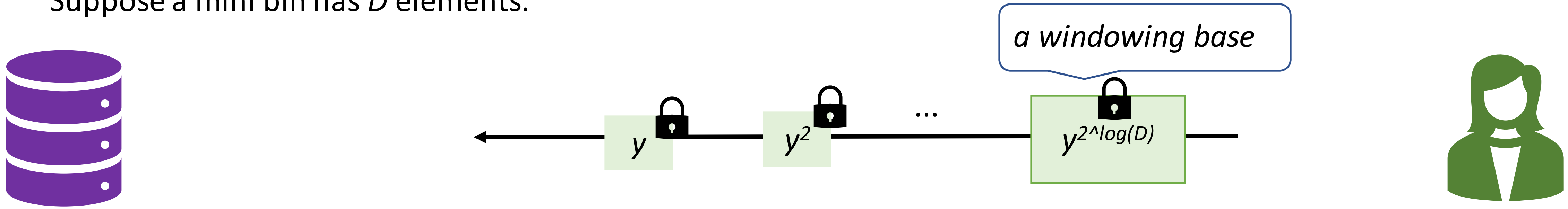


Server recovers all powers: y ... y^D

Client sends log(D) powers.
Can she send less?

Improving windowing: using postage stamp bases

Suppose a mini bin has D elements.



Server recovers all powers: y ... y^D

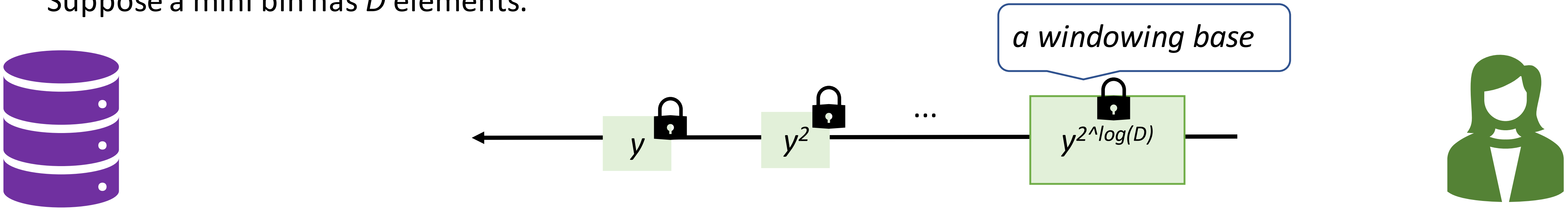
Client sends log(D) powers.
Can she send less?

The global postage stamp problem:
Given h, k integers,
find $1 = a_1 < a_2 < \ldots < a_k$ integers, called *extremal postage stamp basis*,
such that any $1 \leq b \leq D$ can be written as a sum of (at most) h a_i's, with possible repetition, and D is as large as possible.

Example: $h = 4$, $k = 3$: {1, 5, 8} -> can recover any power $D \leq 26$ using circuits of depth ≤2.

Improving windowing: using postage stamp bases

Suppose a mini bin has D elements.



Server recovers all powers: y ... y^D

Client sends log(D) powers.
Can she send less?

The global postage stamp problem:
Given h, k integers,
find $1 = a_1 < a_2 < \ldots < a_k$ integers, called *extremal postage stamp basis*,
such that any $1 \leq b \leq D$ can be written as a sum of (at most) h a_i's, with possible repetition, and D is as large as possible.

Unknown complexity class!
Brute-forced solutions for small instances by Challis and Robinson
https://cs.uwaterloo.ca/journals/JIS/VOL13/Challis/challis6.pdf

Open problems

- Find non-trivial algorithms for computing global postage stamp bases.

- Apply the optimizations for other applications such as:
  - PSI with computation (e.g. Private count of common elements)
  - Private Information Retrieval (PIR)

Thank you!