PIR-PSI: SCALING PRIVATE CONTACT DISCOVERY

PETS 2018

TECHNISCHE UNIVERSITÄT DARMSTADT

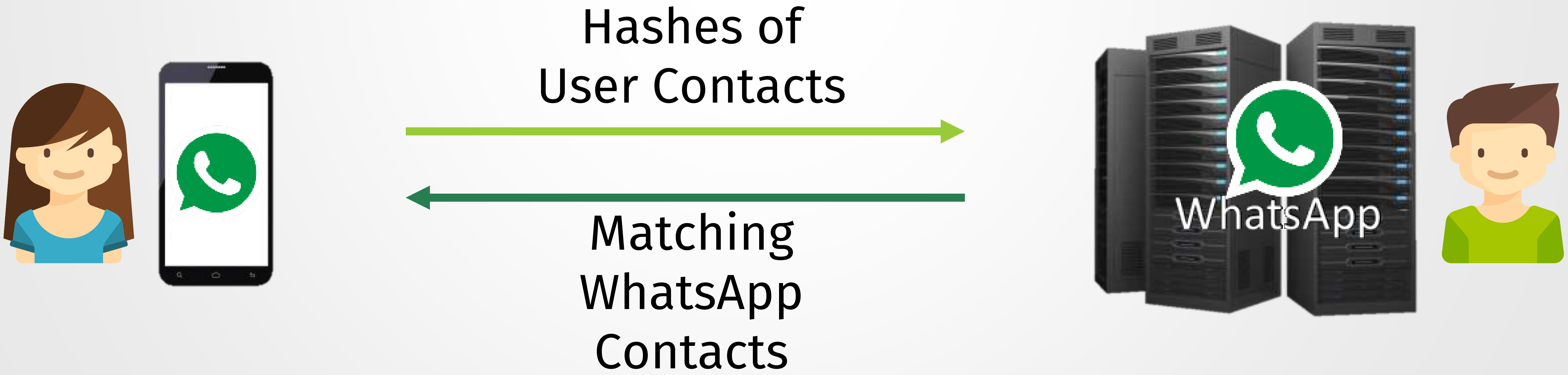
Daniel Demmler

Peter Rindal
Mike Rosulek
Ni Trieu

Oregon State University OSU

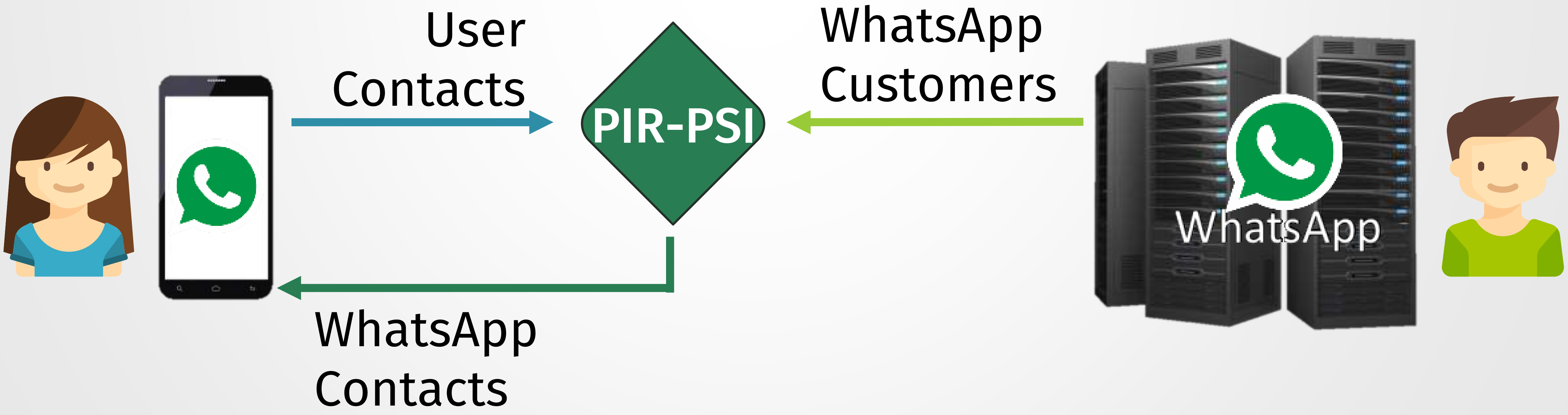
Motivation – Application: Contact Discovery

- Contact discovery tells social network users which of their contacts are in the social network
- An **insecure** naïve hashing-based protocol is used in practice
- Vulnerable to
  - Brute-force attacks (for small input domain, e.g. phone numbers)
  - Comparison with hashes from later sessions

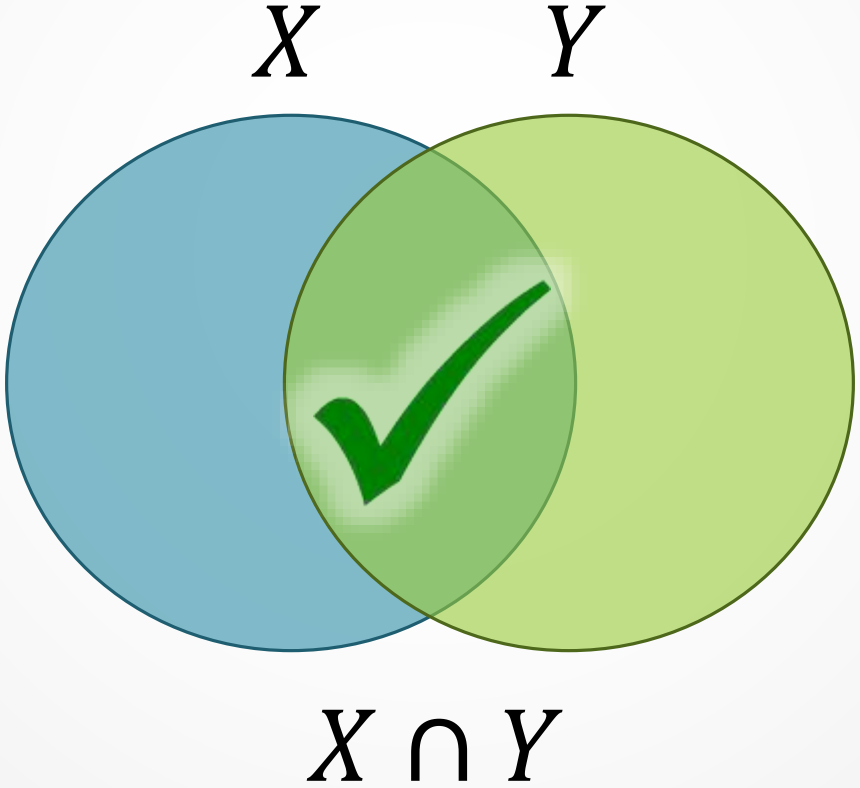


Motivation – Application: Private Contact Discovery

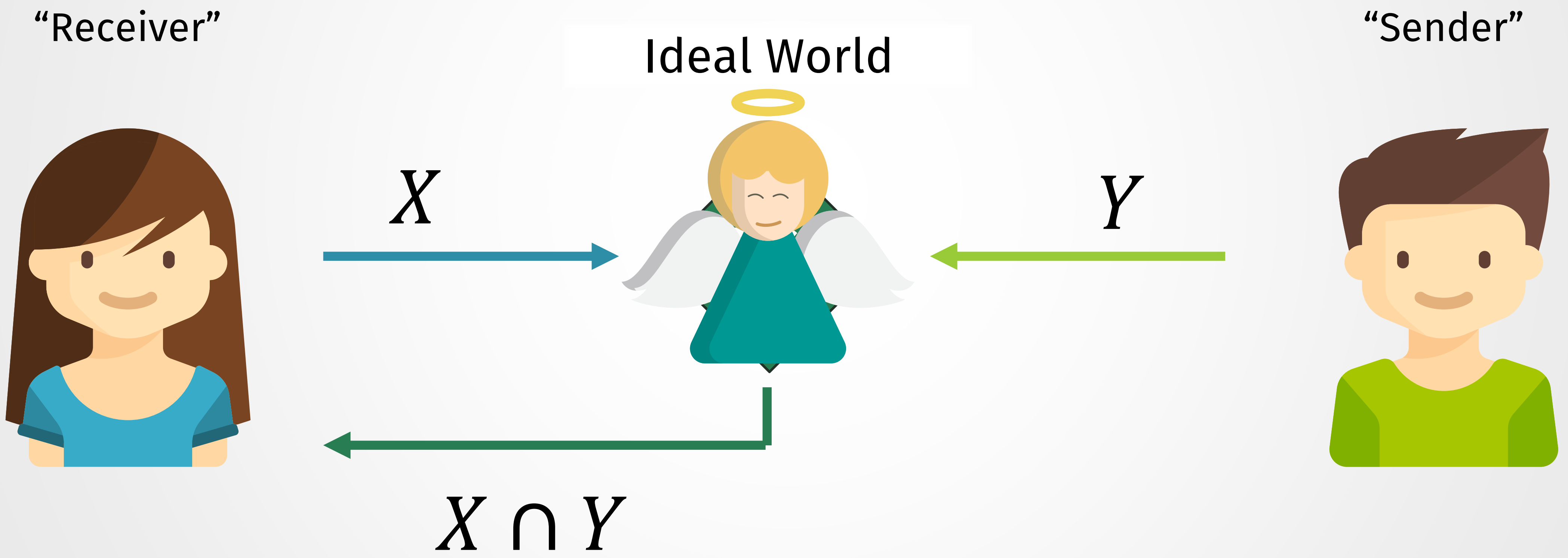
- Contact Discovery should be **efficient** and **scalable**, and **protect the privacy of user inputs**.
- It runs once when a user initially joins a social network
- … and periodically to find contacts that join the social network later on.



Private Set Intersection (PSI)

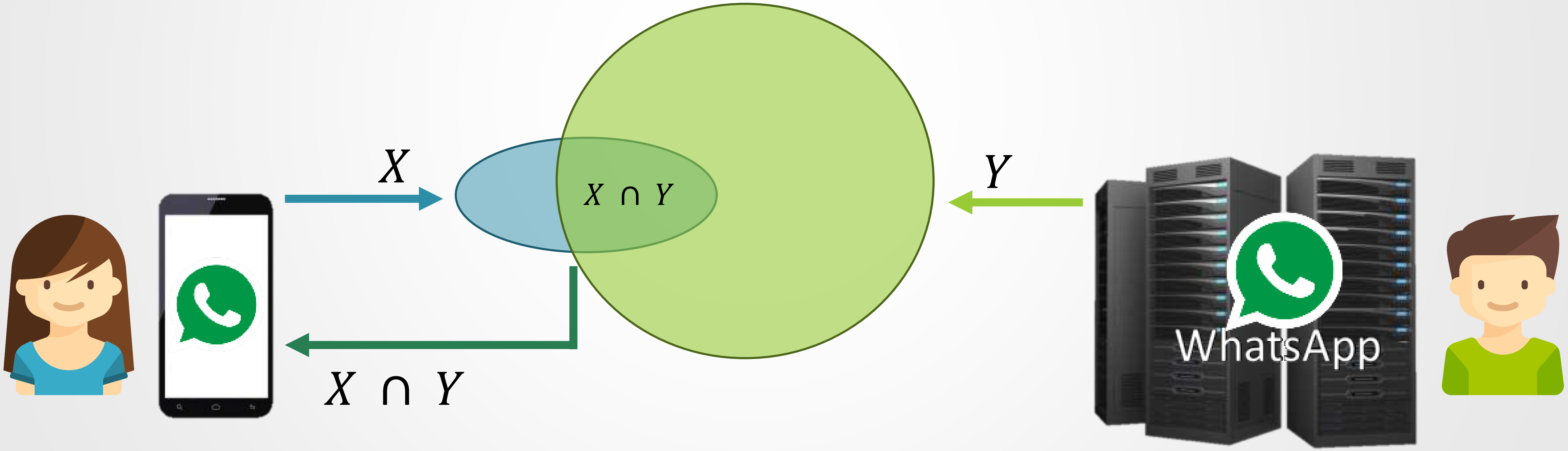


Private Set Intersection (PSI)



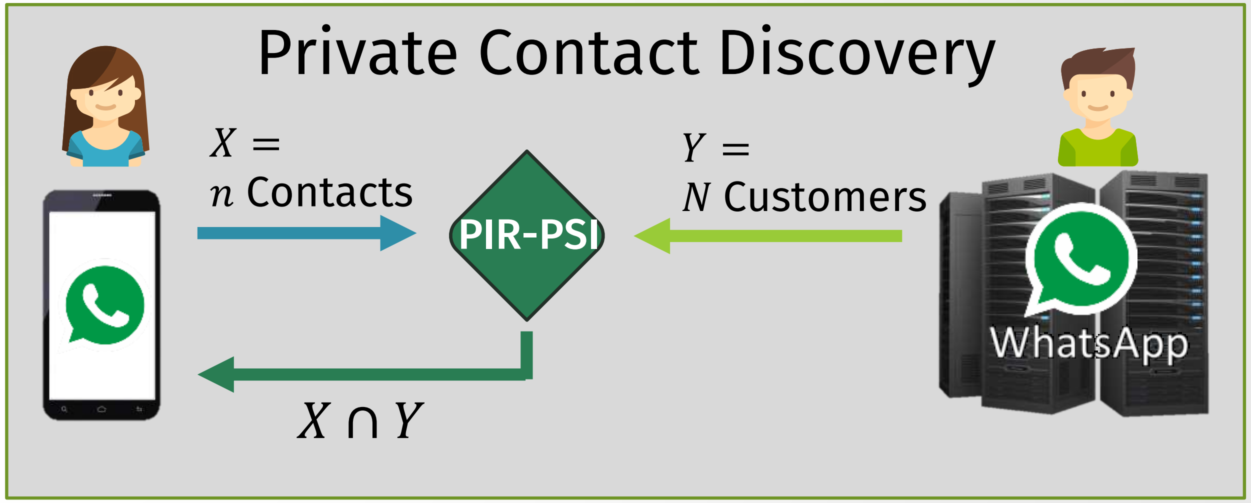
PSI for Contact Discovery

$$|X| = n \ll |Y| = N$$



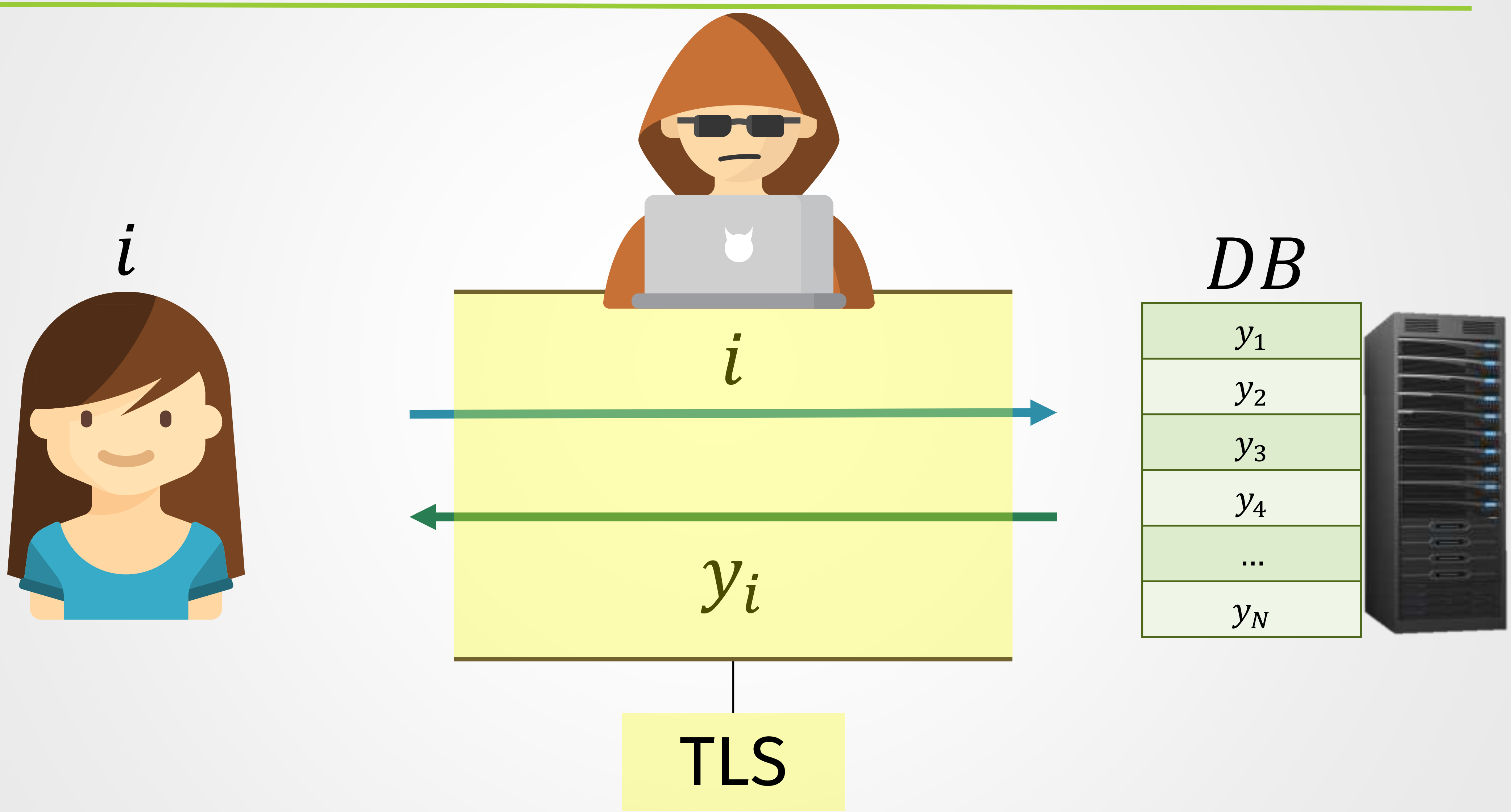
Status-Quo vs. PIR-PSI

- Communication linear in both sets $O(N+n)$
  - What about $N \gg n$?
  - Insecure solution
    - Send small set to other party
    - Communication = $O(\min(N,n))$

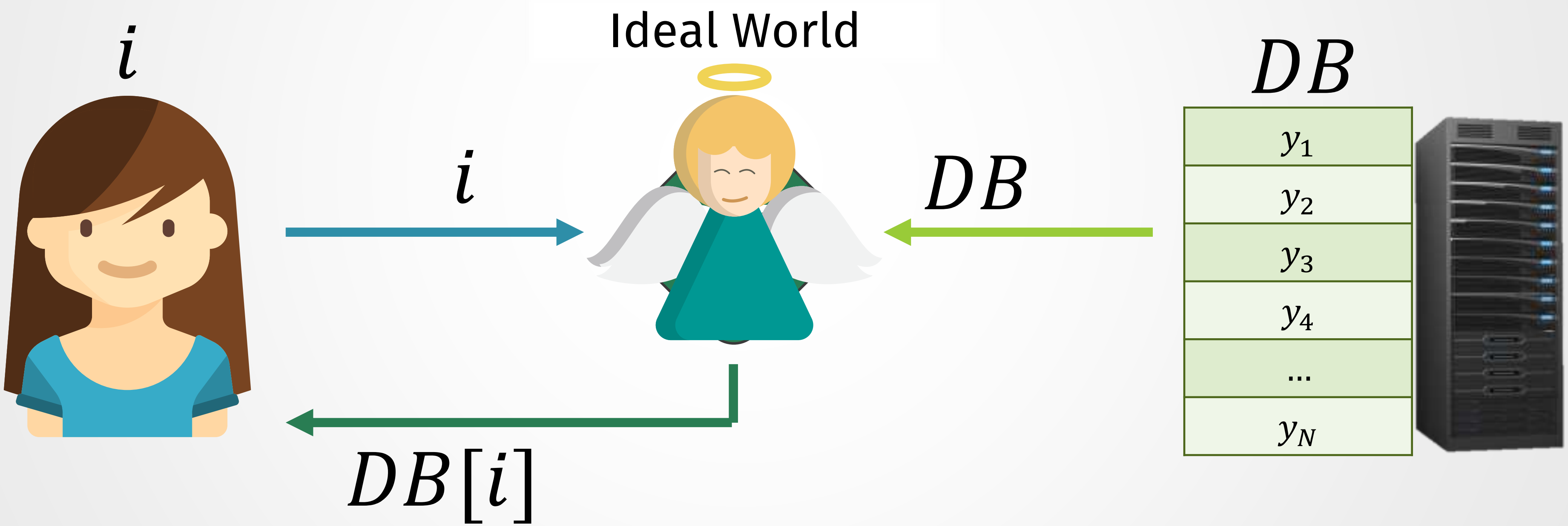


- **PIR-PSI**
  - Communication = $O\left(n \log\left(\frac{N \log n}{n}\right)\right)$
  - Client Computation = $O\left(n \log\left(\frac{N \log n}{n}\right)\right)$ AES operations
  - Server Computation = $O(N \log n)$ AES operations

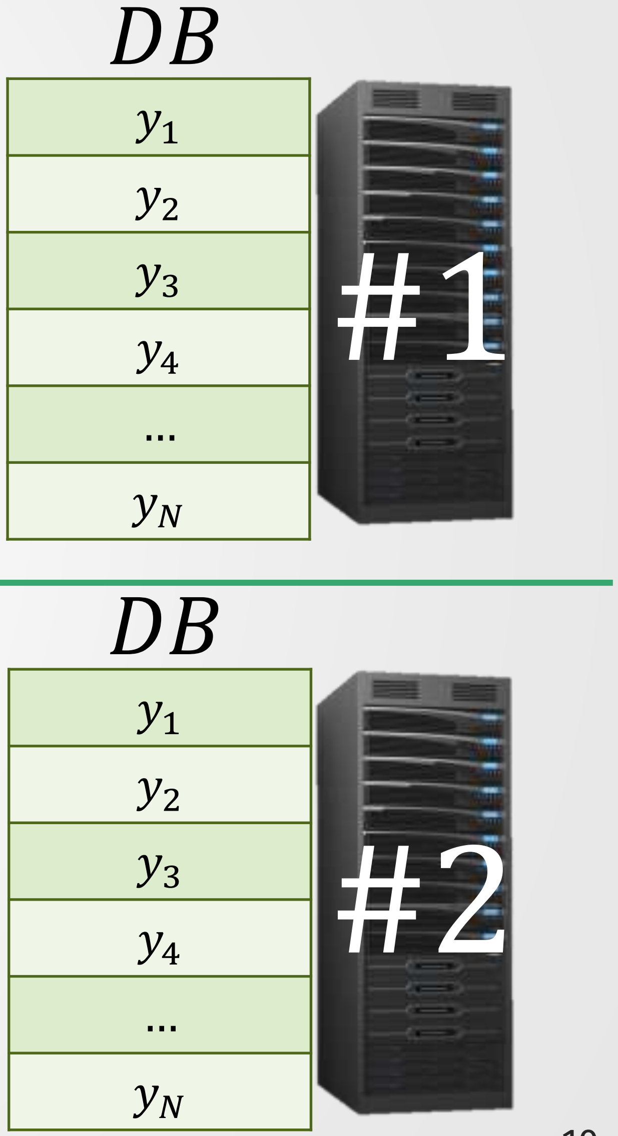
Plaintext Database Query



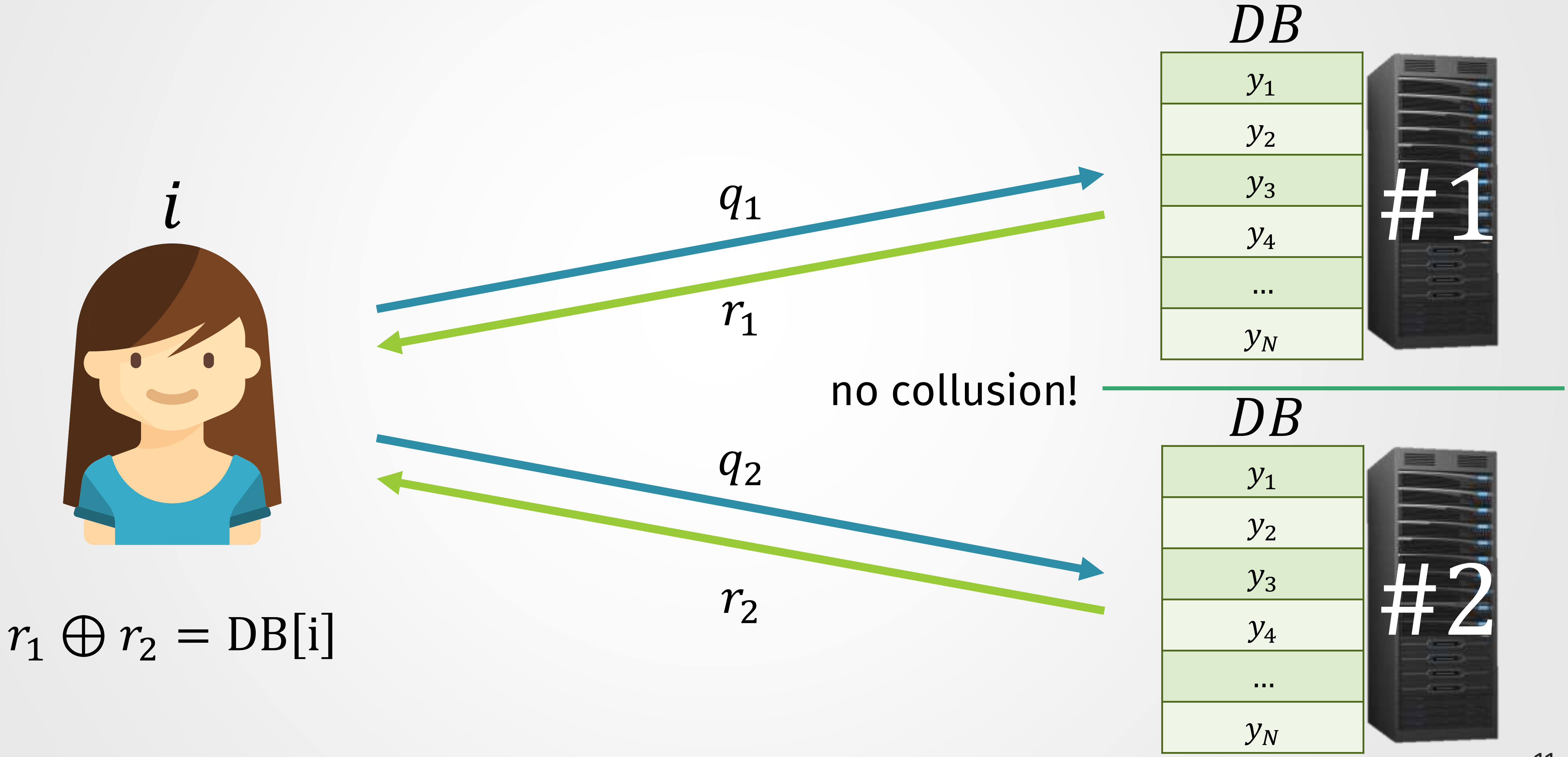
Private Information Retrieval (PIR)



2-Server PIR [CGKS95]



2-Server PIR [CGKS95]



Example: 2-Server Linear Summation PIR [CGKS95]

$i = 2 \Rightarrow\ q = 001\,000$

q_1 chosen at random

$q_2 = q \oplus q_1$

Query length $|q| = N$

$q_1 = 010\,001$

$r_1 = DB[1] \oplus DB[5]$

$r_i = q_i \cdot DB$

$q_2 = 011\,001$

$r_2 = DB[1] \oplus DB[2] \oplus DB[5]$

$r_1 \oplus r_2 = \mathrm{DB}[2]$

DB

y_1
y_2
y_3
y_4
…
y_N

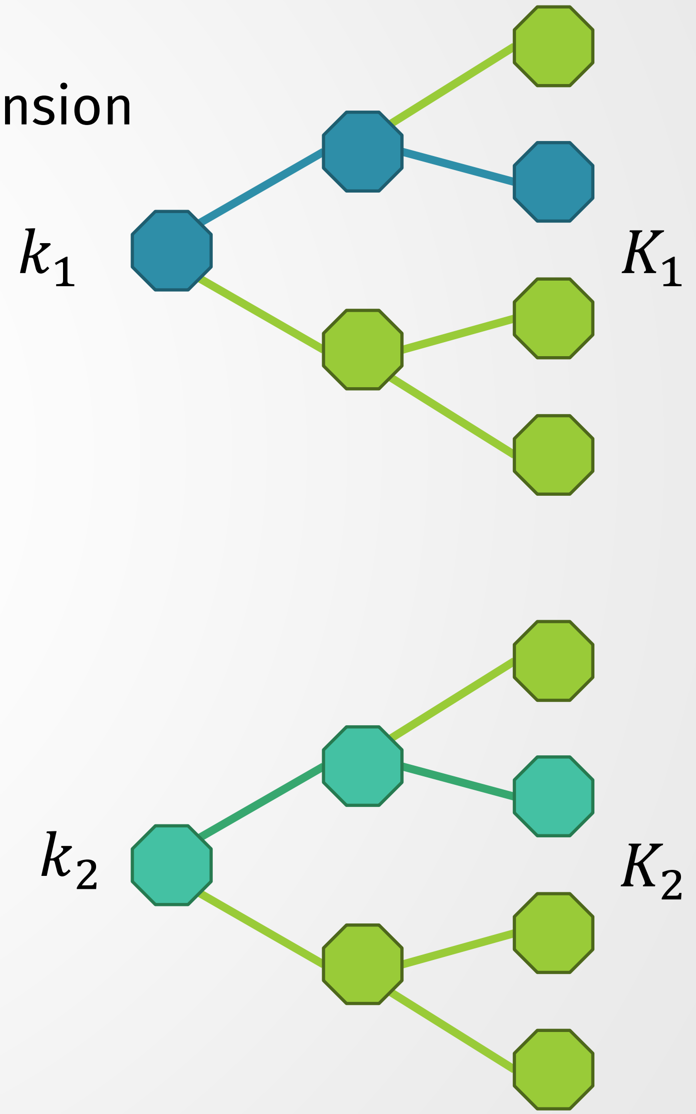
DB

y_1
y_2
y_3
y_4
…
y_N

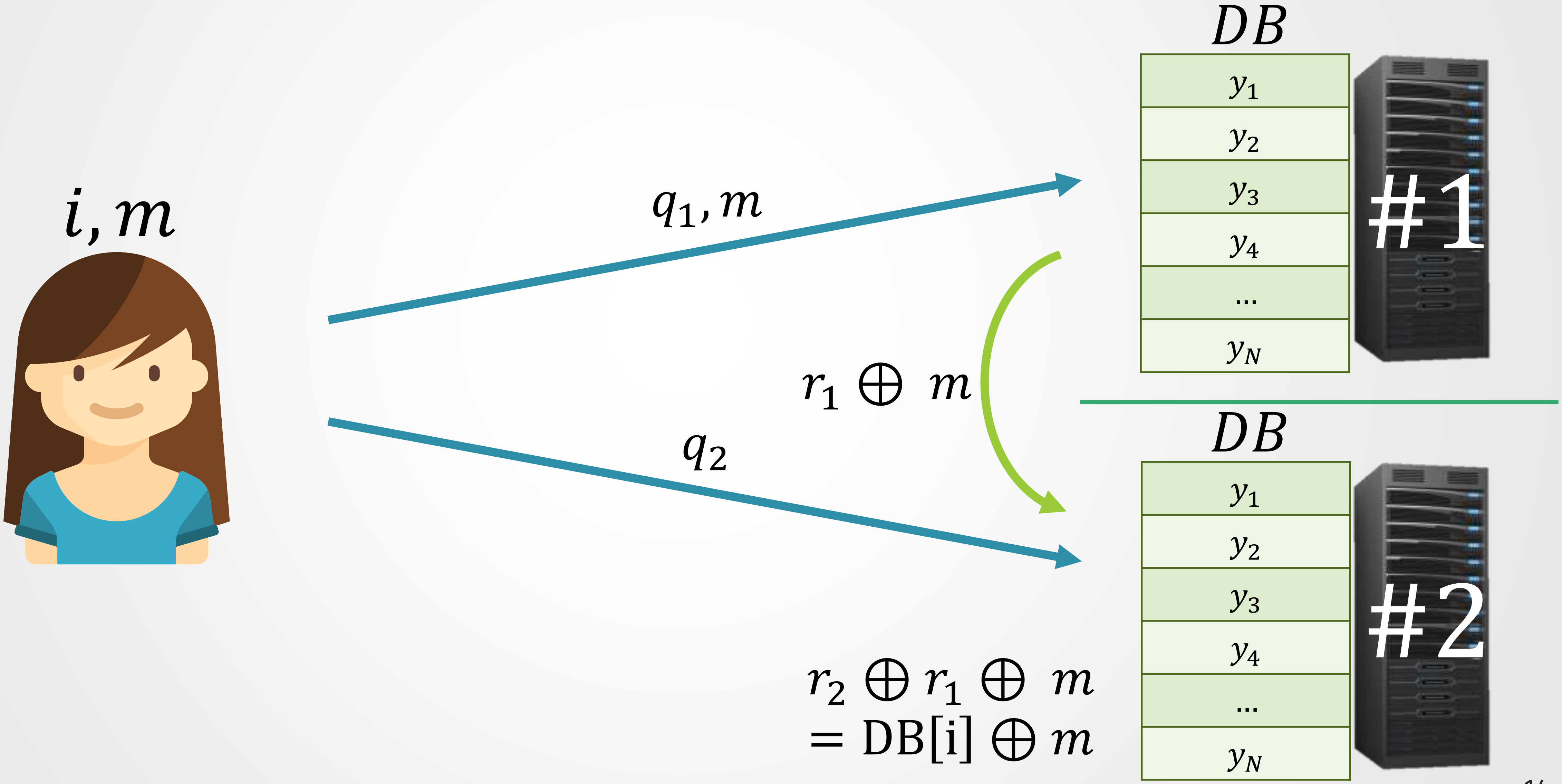
PIR from Distributed Point Functions (DPFs)

- Point Functions: $PF = \{f_{i,v}: f_{i,v}(i) = v, f_{i,v}(x) = 0 \; \forall \; x \neq i\}$.
- **Distributed** PFs allow 2 parties the secret-shared PF evaluation, without revealing i, v.
- DPFs are described by short *keys* k_1, k_2 of length $O(\log N)$, where N is the domain of i.
- By using $v = 1$, i.e., a DPF returning 1 only at index i, we can express the plain text query q and thus build 2-server PIR with $O(\log N)$ communication complexity.
- Instantiated efficiently with AES.

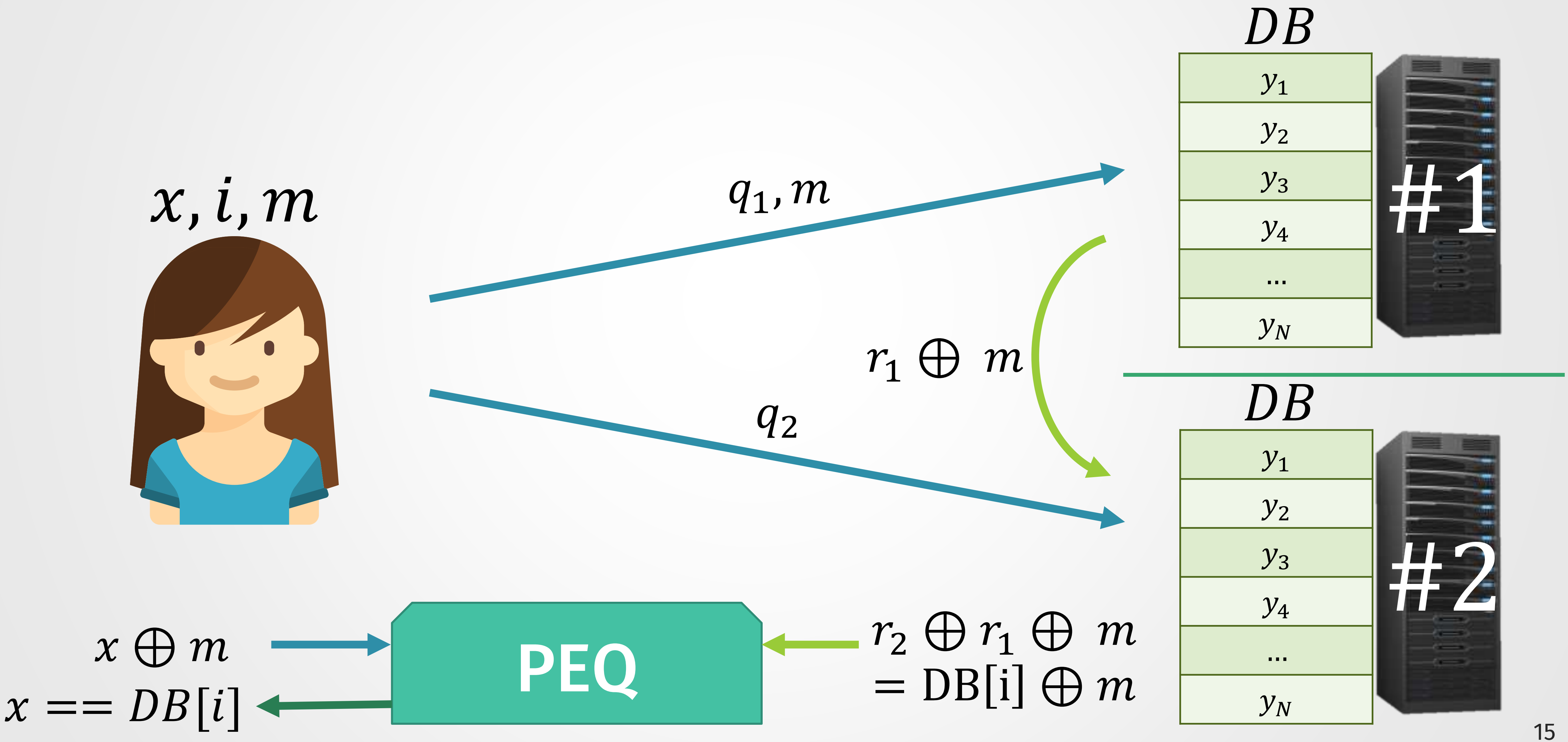
Intuition:
DPF Key Expansion



Designated-Output PIR

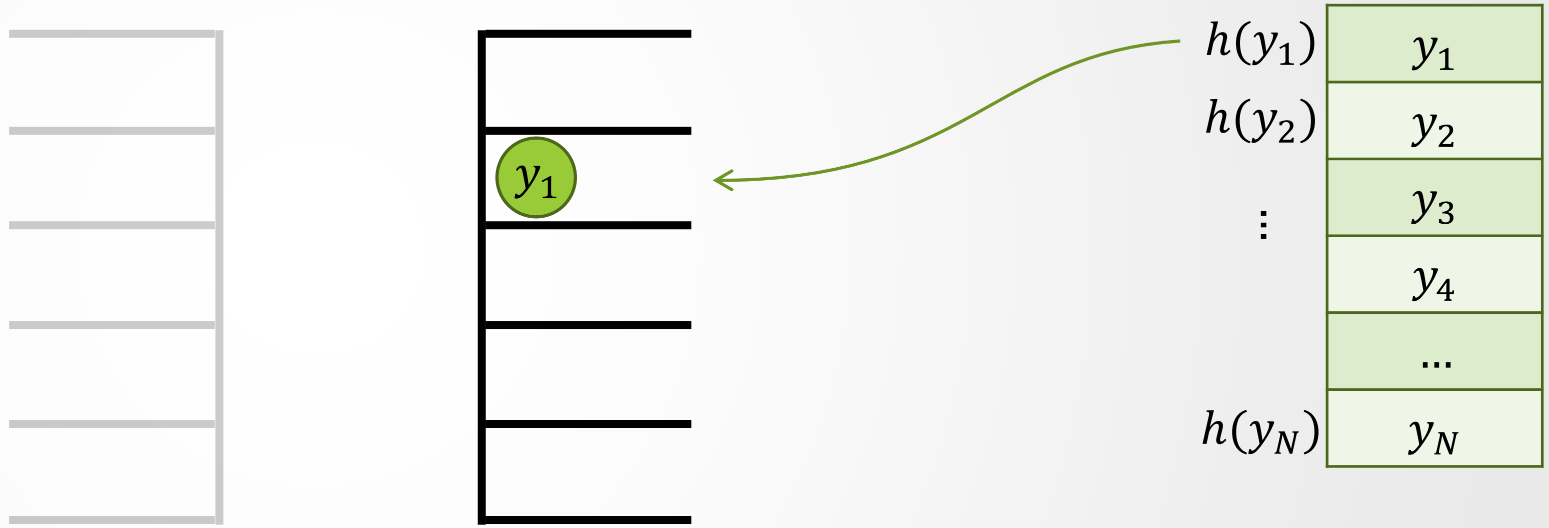


PIR Private Equality Test



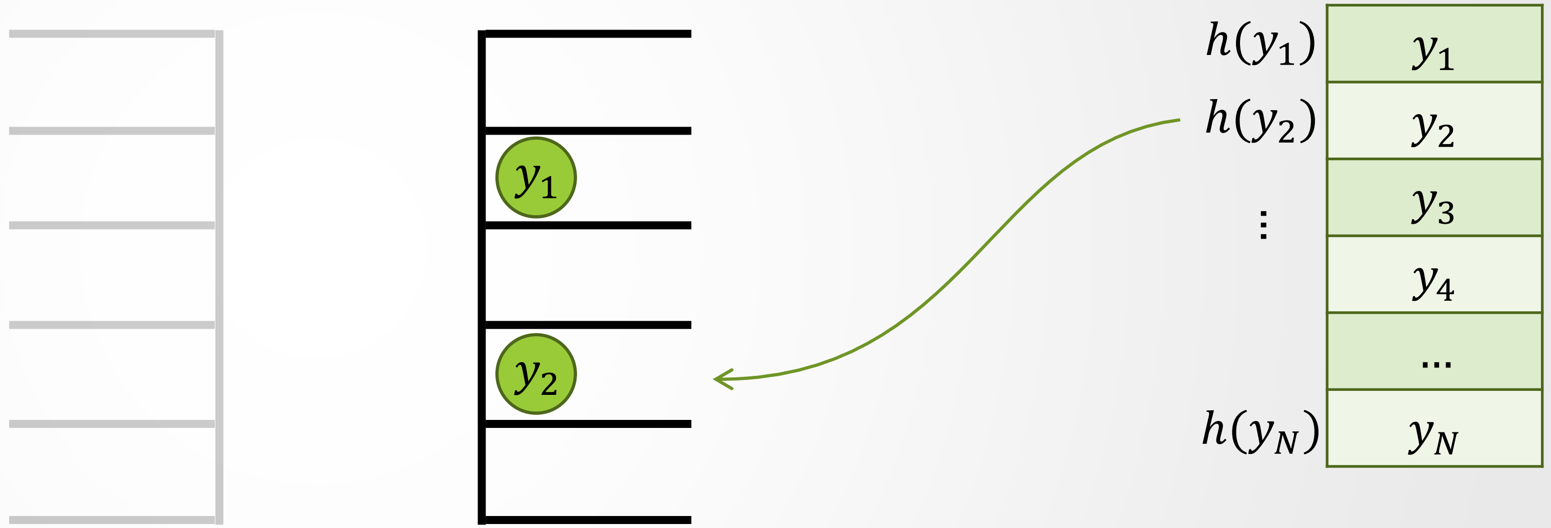
Cuckoo Hashing

- Server performs Cuckoo hashing.



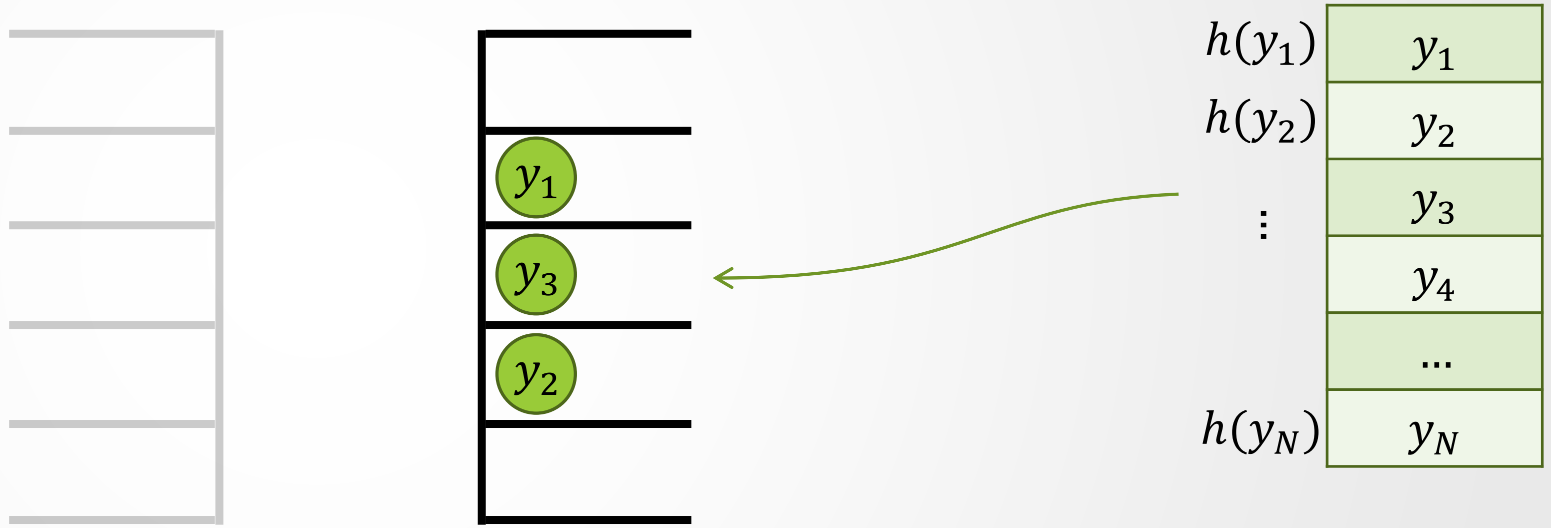
Cuckoo Hashing

- Server performs Cuckoo hashing.



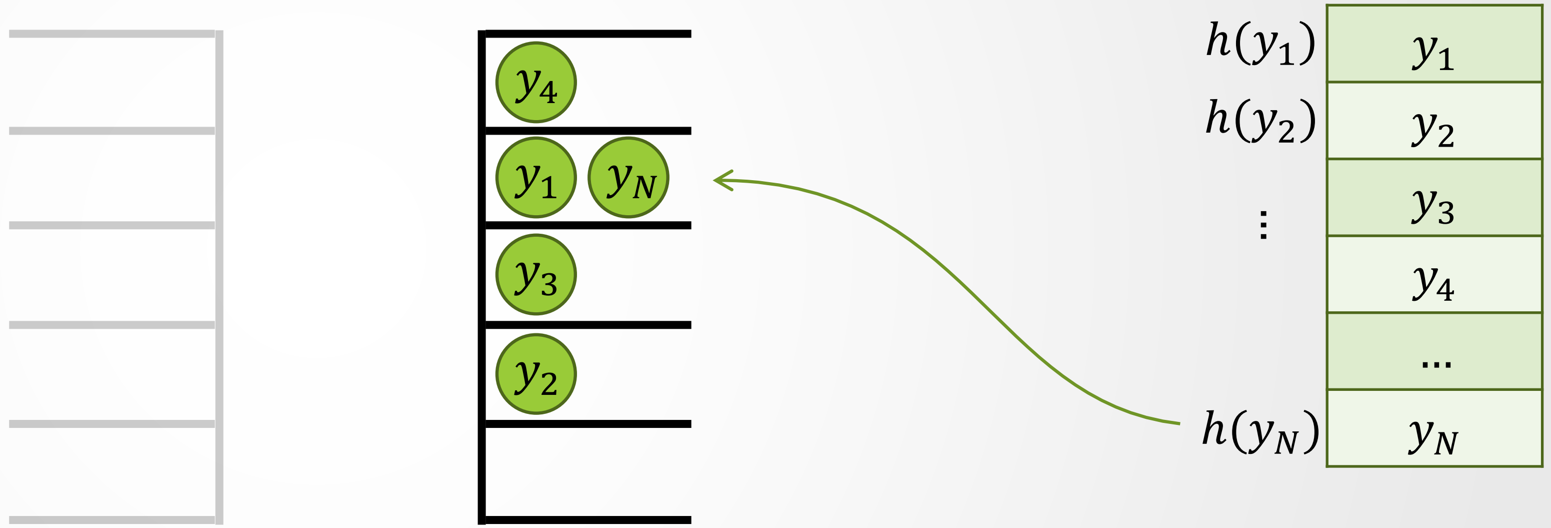
Cuckoo Hashing

- Server performs Cuckoo hashing.



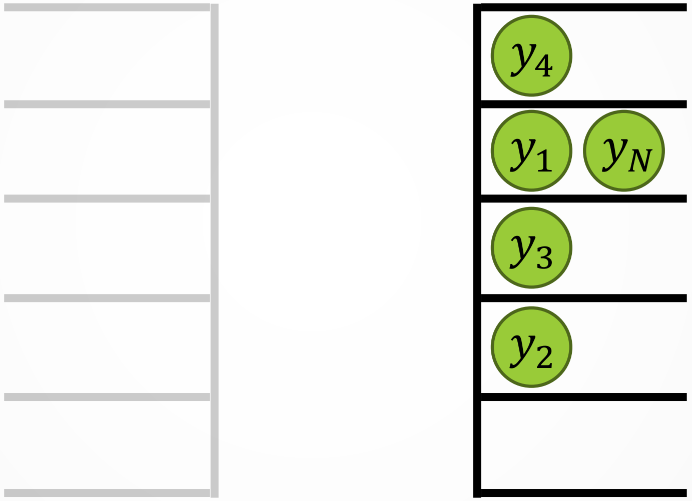
Cuckoo Hashing

- Server performs Cuckoo hashing.



Cuckoo Hashing

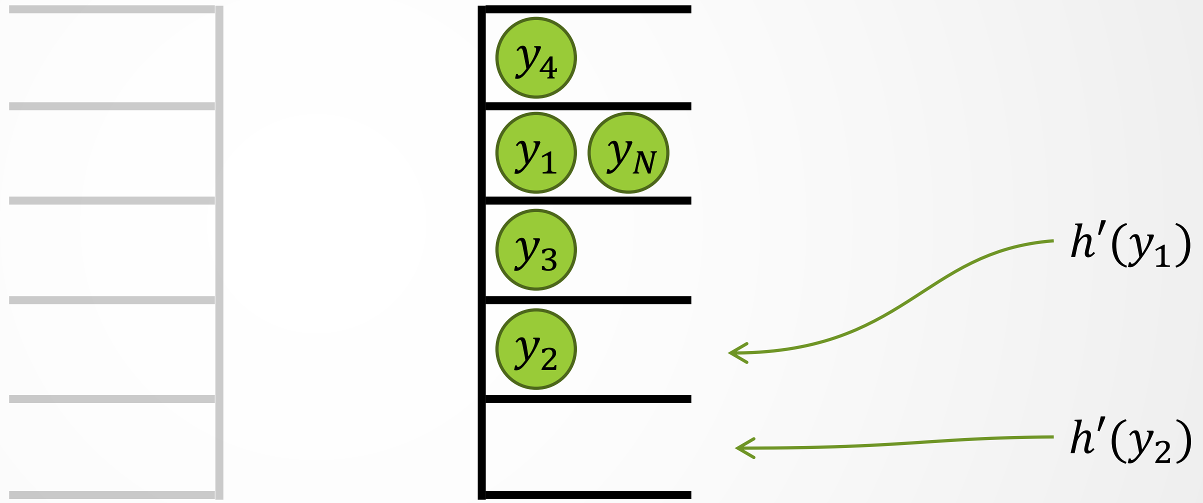
- Server performs Cuckoo hashing.



$\leftarrow$ Collision: $h(y_1) = h(y_N)$

Cuckoo Hashing

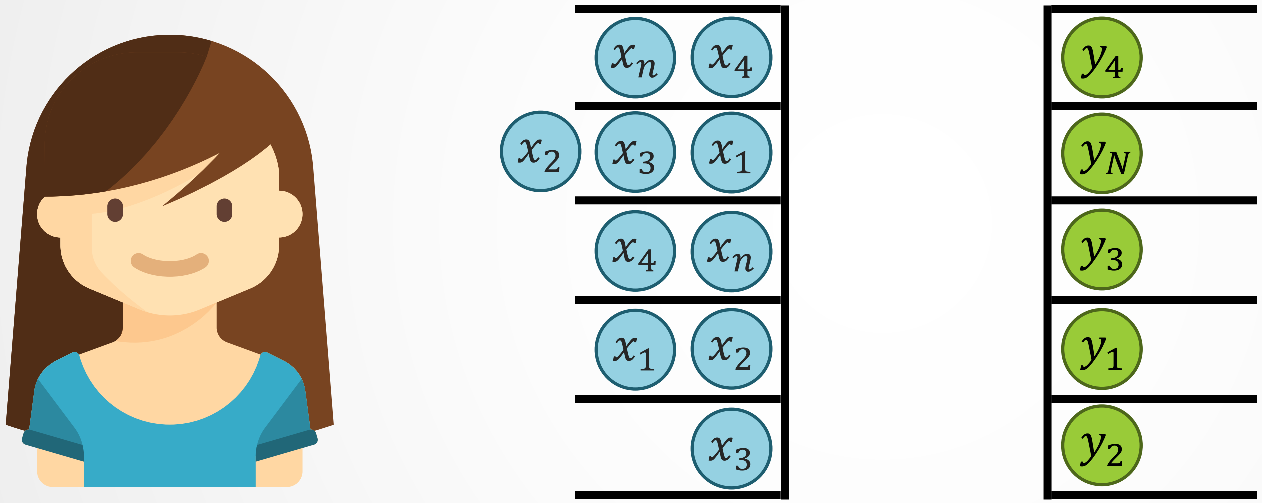
- Server performs Cuckoo hashing.



- To avoid collisions: use multiple hash functions - in this example: h, h'.
- In our implementation we used 3 hash functions and a cuckoo expansion factor of $e \approx 1.4$ for a cuckoo failure probability of 2^{-20} during one-time initialization.

Cuckoo Hashing

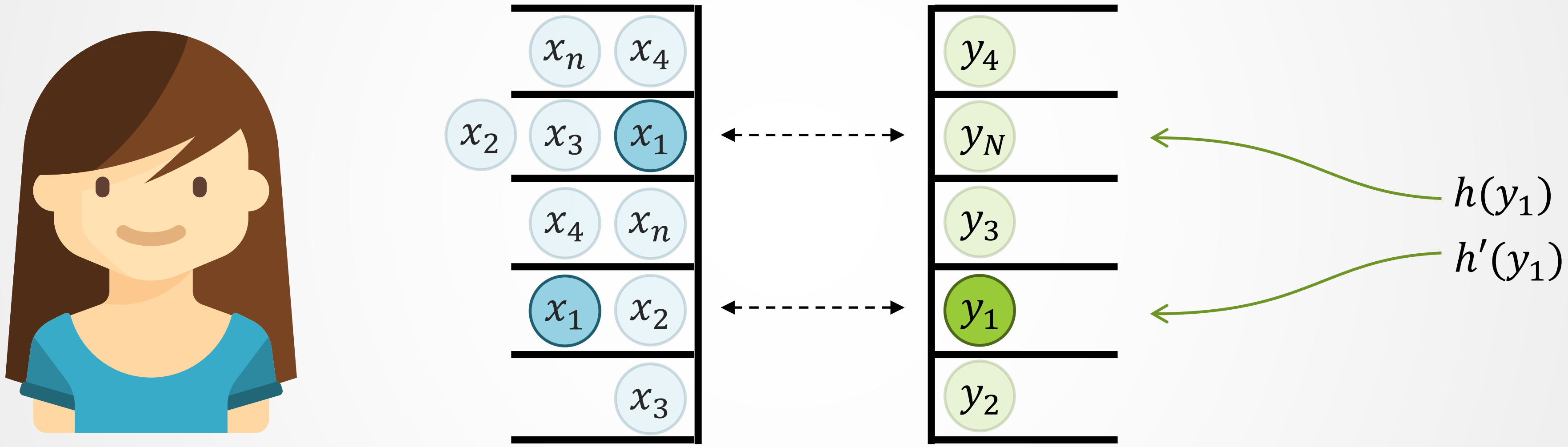
- Every element can be located in two possible bins.



- The client computes all hash functions for every element.

Cuckoo Hashing

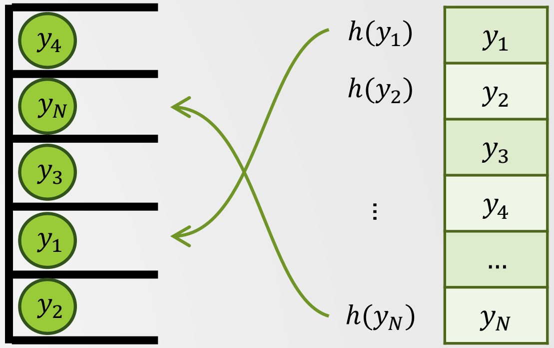
- Every element can be located in two possible bins.



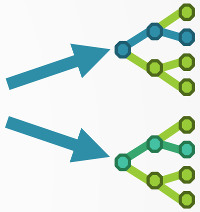
- To check if the server holds x_1, the client runs a PIR-PEQ with the 2nd and 4th bin.
- In the full protocol: instead of single PIR-PEQ, we run all of them together in a **PSI protocol**.

PIR-PSI Overview

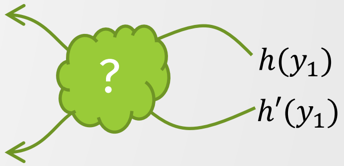
1. **Cuckoo Hashing**
   - Both servers compute the same cuckoo hash table for their N elements.



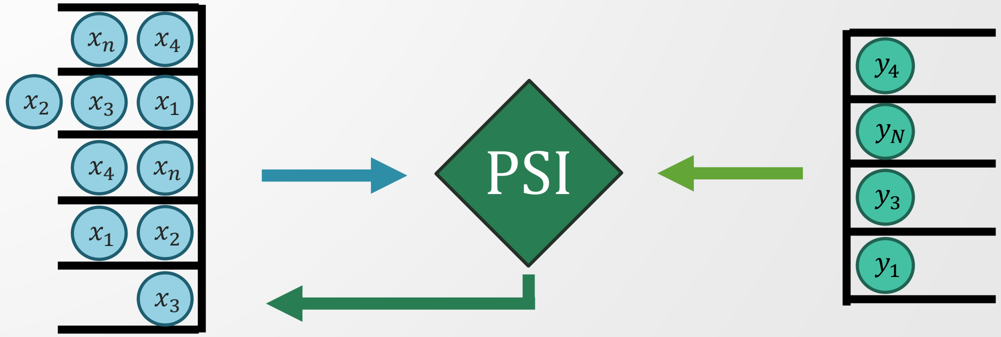
2. **DPF-PIR Query**
   - The client delegated extraction of n elements from the cuckoo table.



3. **Oblivious Shuffle**
   - One server receives the other server's masked output and obliviously shuffles the PIR results to hide which Cuckoo hash function was used.

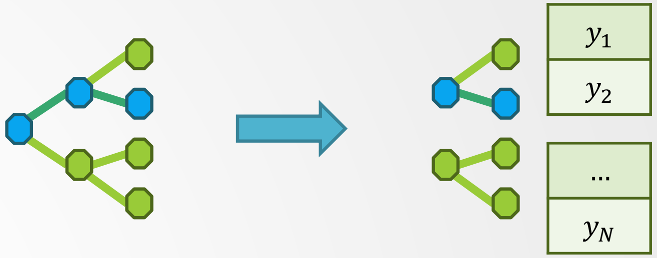


4. **Small PSI**
   - A standard PSI protocol is used to determine intersection.

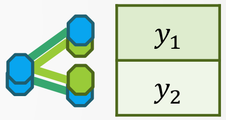


Optimizations

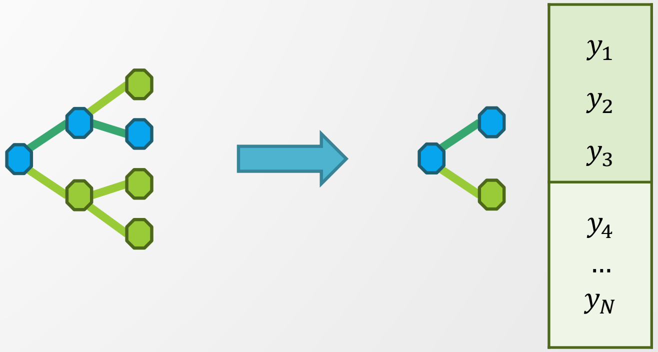
- **Binning**
  - Instead of running full domain DPFs, we partition the server cuckoo table into bins and a smaller DPFs per bin.
  - Parallelization!



- **Batching**
  - Instead of running DPF queries separately,
    run all queries in each bin in parallel.
  - Only a single pass over the cuckoo table for multiple queries.

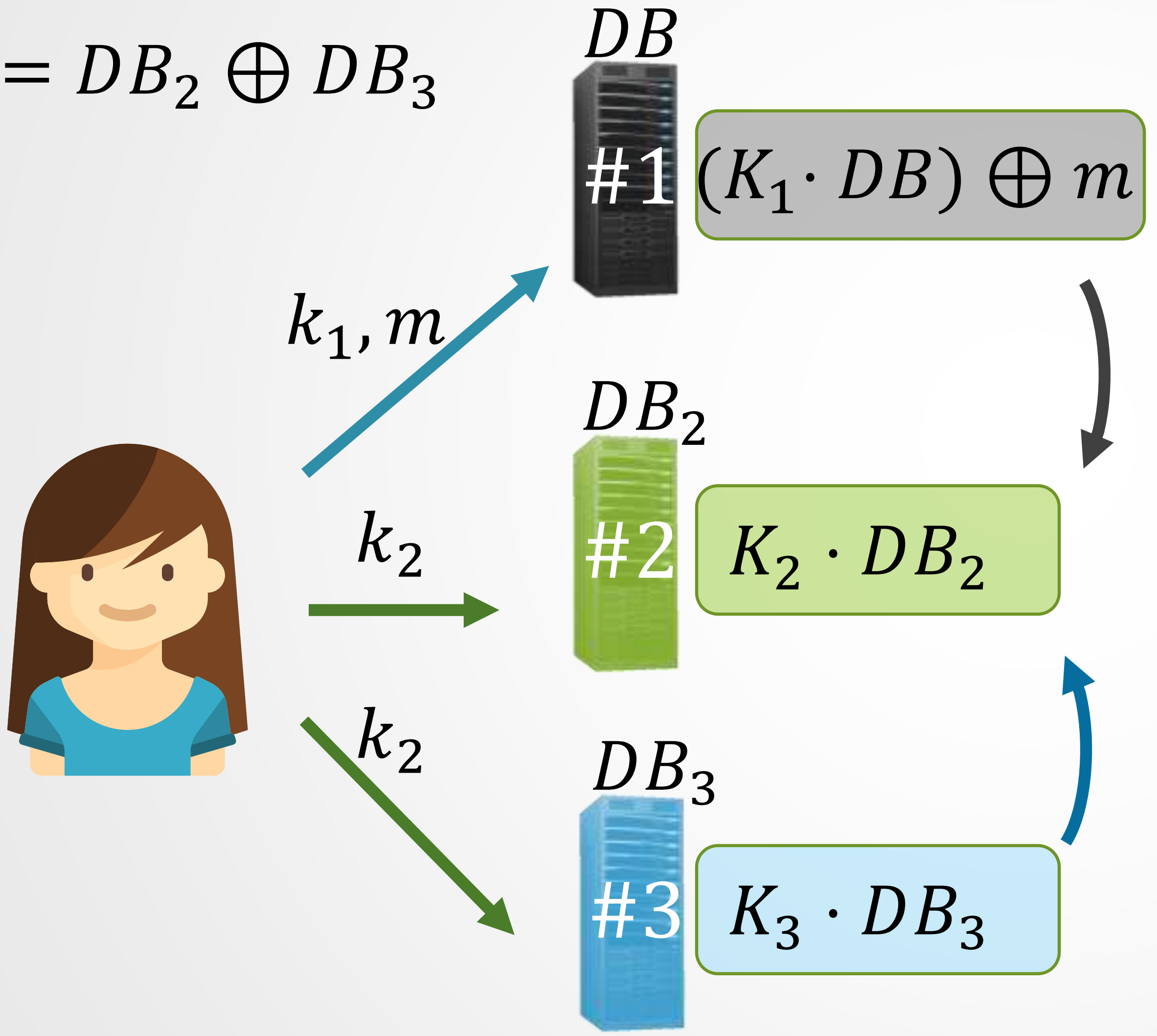


- **Larger PIR Blocks**
  - PIR queries can return multiple cuckoo table entries.
  - less communication, more computation in PSI.



PIR-PSI with 3 PIR Servers

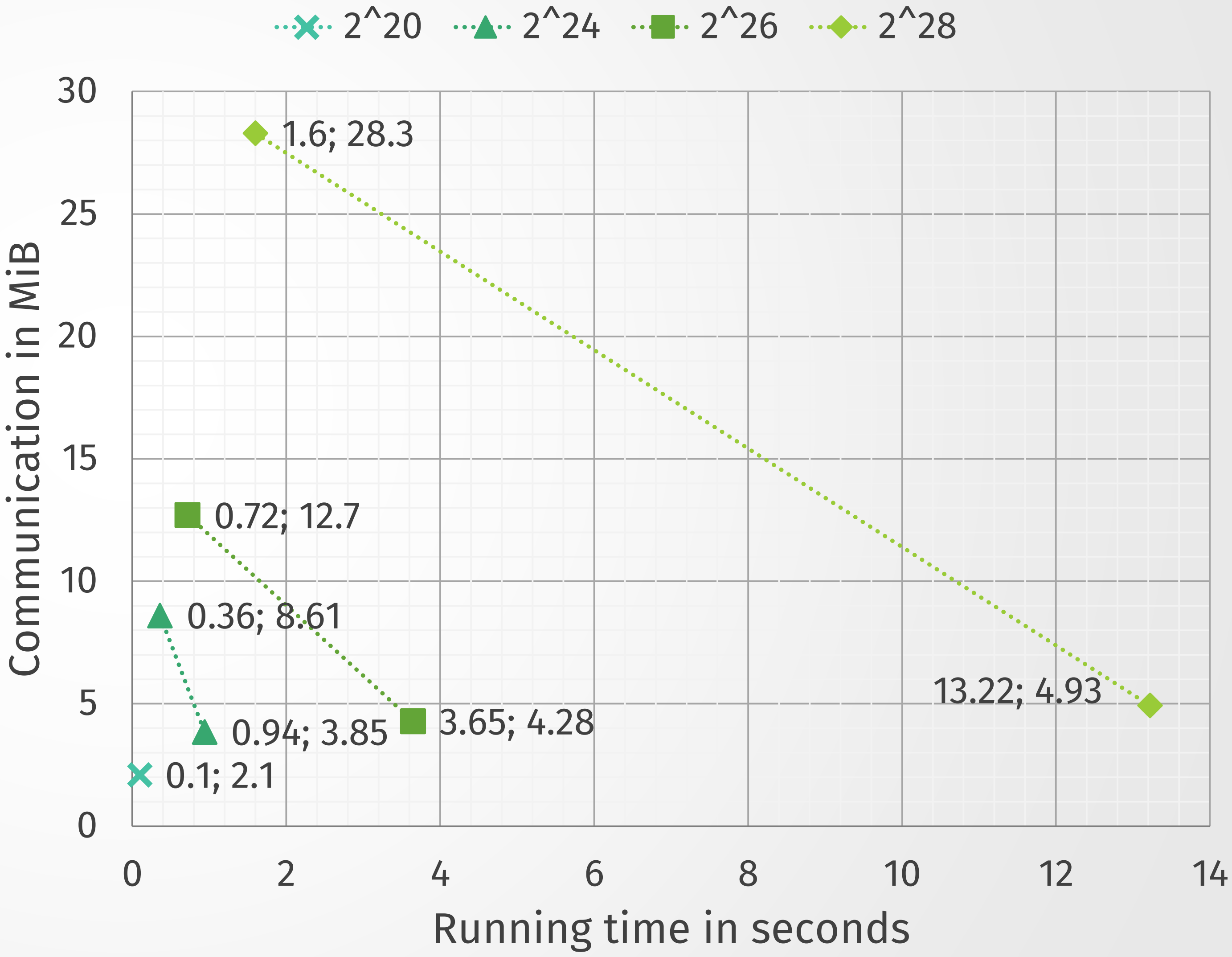
$DB = DB_2 \oplus DB_3$



$$(K_2 \cdot DB_2) \oplus (K_2 \cdot DB_3) \oplus (K_1 \cdot DB) \oplus m$$
$$=$$
$$K_2 \cdot (DB_2 \oplus DB_3) \oplus (K_1 \cdot DB) \oplus m$$
$$=$$
$$K_2 \cdot DB \oplus K_1 \cdot DB \oplus m$$
$$=$$
$$DB[i] \oplus m$$

PIR-PSI Performance

- Communication and running time for $n = 1\,024$ client elements and server set sizes $N \in \{2^{20}, 2^{24}, 2^{26}, 2^{28}\}$.
- Benchmarked in Gigabit LAN, on 1 machine with 36 x 2.3 GHz. Implementation set to use 4 threads.
- Client computation is $\approx 10\%$ of total.
- Parameters for communication / computation trade-off



Conclusion

- Combination of DPF-based PIR with state-of the art PSI to achieve scalable contact discovery.
- Efficient open-source C++ implementation on Github: `github.com/osu-crypto/libPSI`
- Many more details in the paper!
  - Security Analysis
  - Cuckoo Hashing Parameters
  - Detailed performance analysis and comparison with related work
  - Extensions

Thank you!

TECHNISCHE UNIVERSITÄT DARMSTADT

Daniel Demmler

Peter Rindal
Mike Rosulek
Ni Trieu

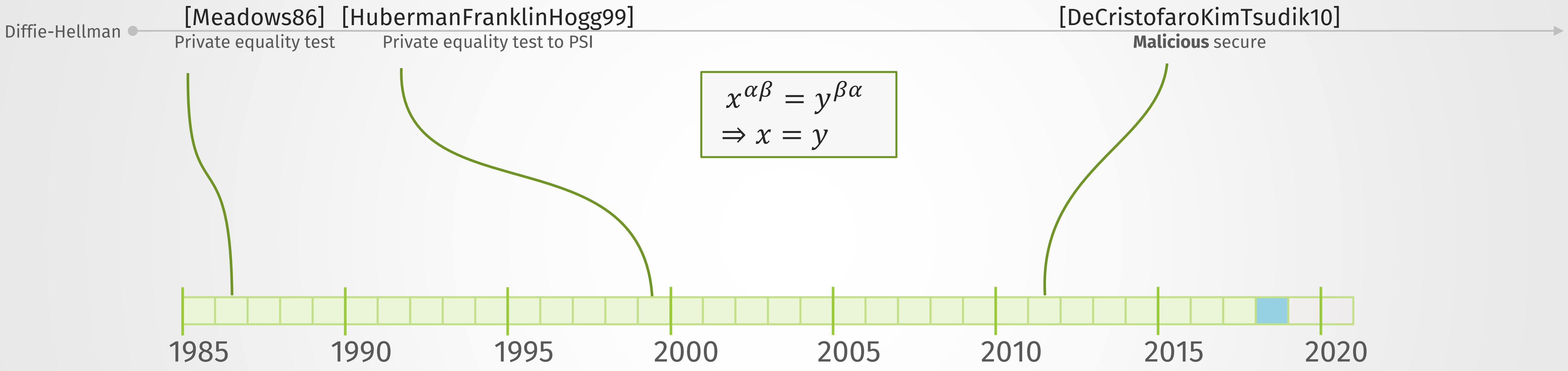
Oregon State UNIVERSITY OSU

References

- Some icons are made by Freepik from flaticon.com

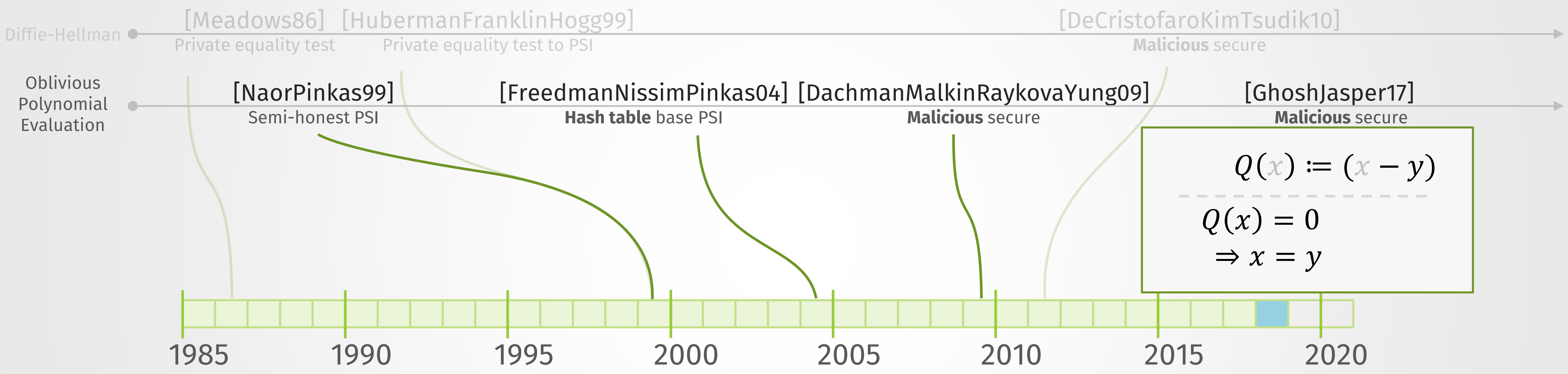
- Extra / Backup slides coming up next...

A *Sampling* of PSI Over the Decades



$$\boxed{\begin{array}{l} x^{\alpha\beta} = y^{\beta\alpha} \\ \Rightarrow x = y \end{array}}$$

A *Sampling* of PSI Over the Decades



A *Sampling* of PSI Over the Decades

Diffie-Hellman

[Meadows86]
Private equality test

[HubermanFranklinHogg99]
Private equality test to PSI

[DeCristofaroKimTsudik10]
Malicious secure

Oblivious Polynomial Evaluation

[NaorPinkas99]
Semi-honest PSI

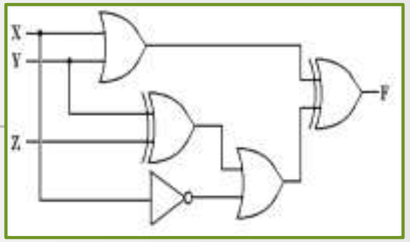
[FreedmanNissimPinkas04]
Hash table base PSI

[DachmanMalkinRaykovaYung09]
Malicious secure

[GhoshJasper17]
Malicious secure

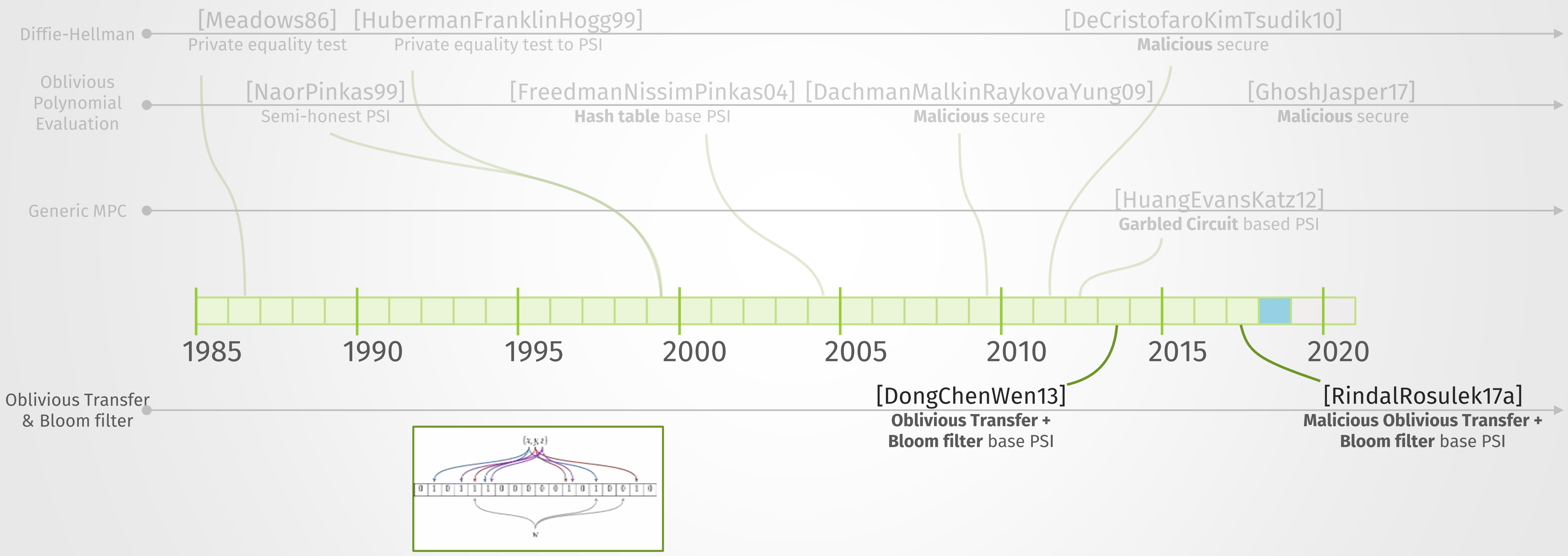
Generic MPC

[HuangEvansKatz12]
Garbled Circuit based PSI

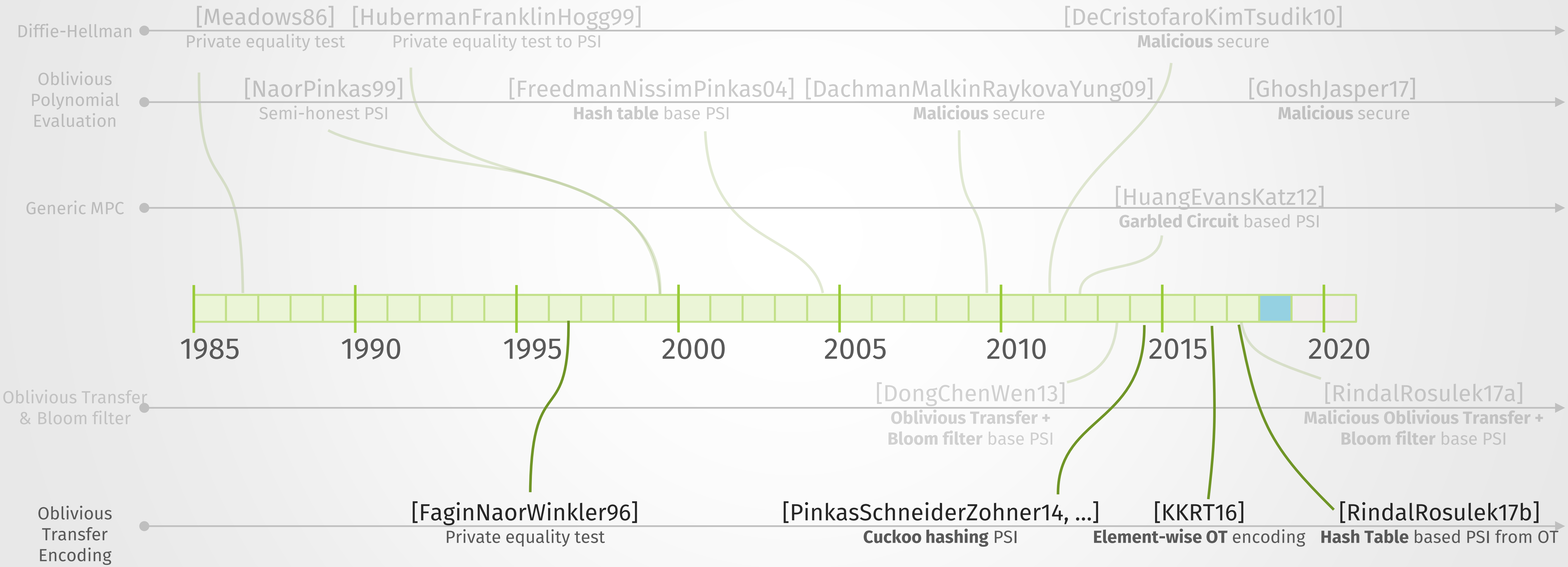


1985 1990 1995 2000 2005 2010 2015 2020

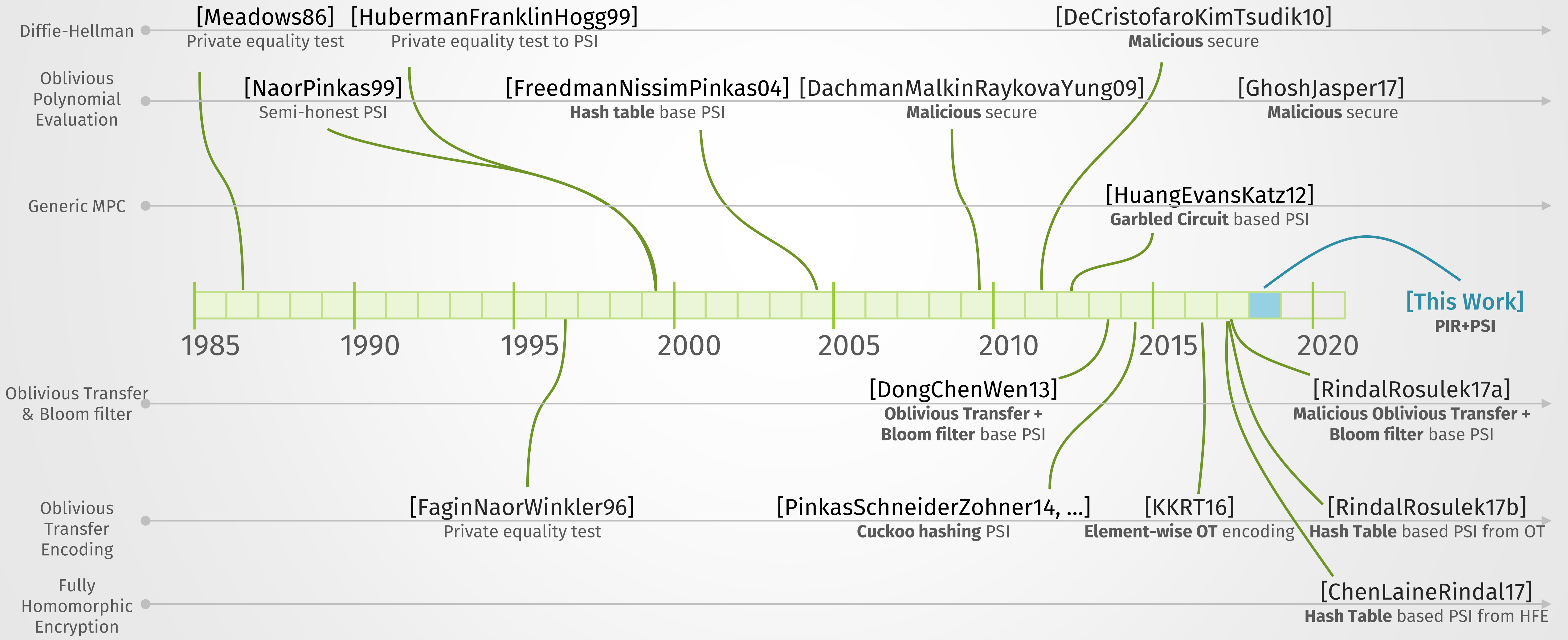
A *Sampling* of PSI Over the Decades



A *Sampling* of PSI Over the Decades



A *Sampling* of PSI Over the Decades



A *Sampling* of PSI Over the Decades