Waldo: A Private Time-Series Database from Function Secret Sharing

Emma Dauterman, Mayank Rathee, Raluca Ada Popa, and Ion Stoica
UC Berkeley

IEEE Security & Privacy 2022

Time-series data

Growing trend: Rely on cloud databases to store and query time-series data

Challenge: Data is often sensitive

Danger of server compromise

Smart home

Industrial IoT

Remote patient monitoring

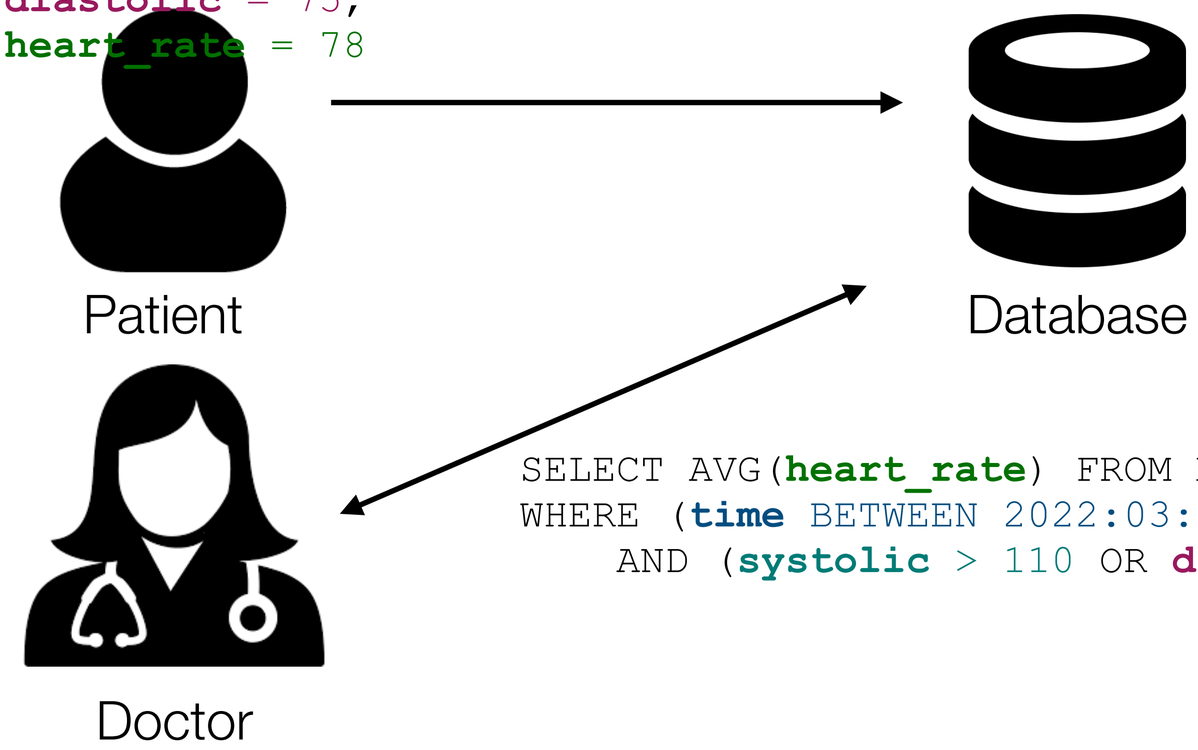
Time-series data: remote patient monitoring

```
time = 2022:03:00,
systolic = 110,
diastolic = 70,
heart_rate = 80
```

time	systolic	diastolic	heart_rate
2022:03:00	110	70	80
2022:03:01	112	68	72
2022:03:02	108	72	76

Time-series data: remote patient monitoring

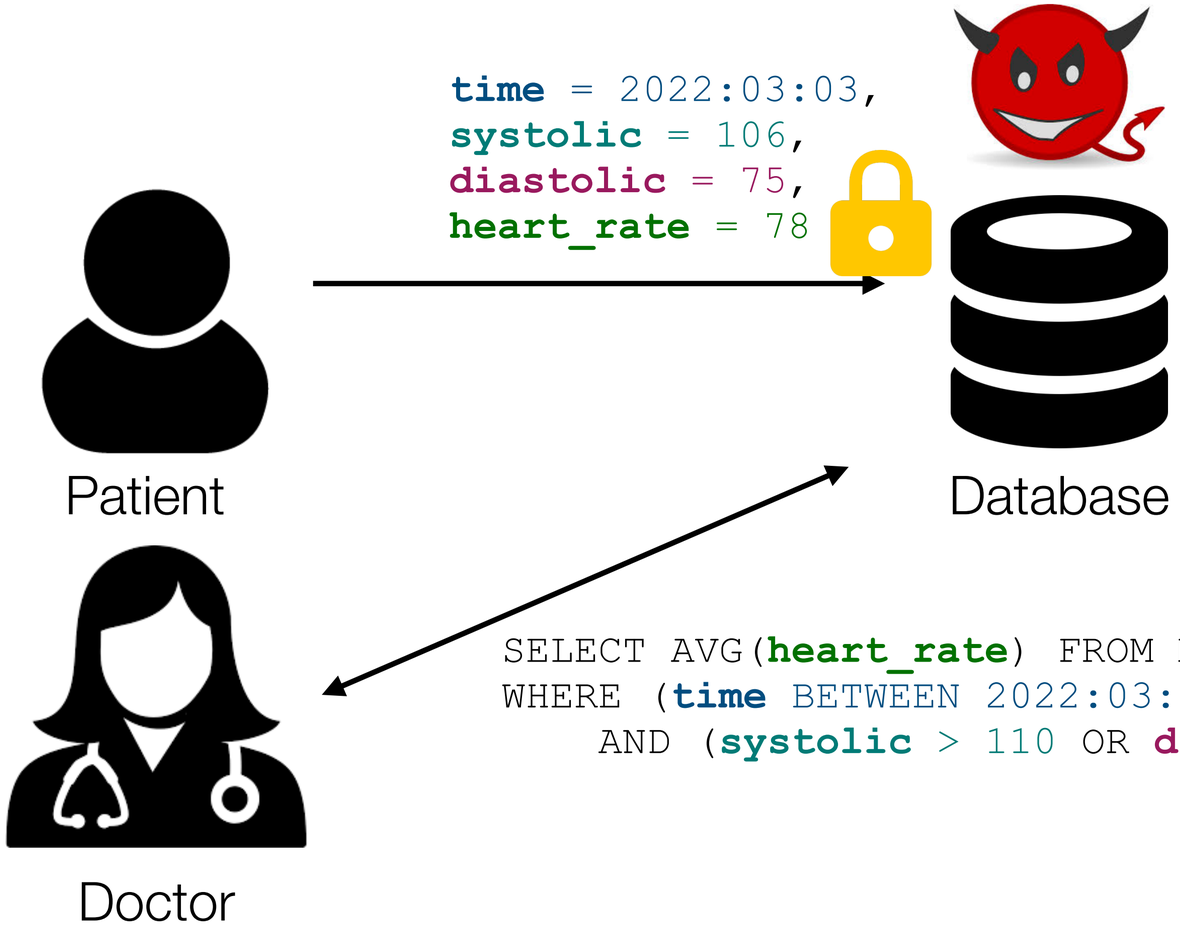
```
time = 2022:03:03,
systolic = 106,
diastolic = 75,
heart_rate = 78
```



time	systolic	diastolic	heart_rate
2022:03:00	110	70	80
2022:03:01	112	68	72
2022:03:02	108	72	76

```
SELECT AVG(heart_rate) FROM MedicalHistory
WHERE (time BETWEEN 2022:03:01 and 2022:03:02)
    AND (systolic > 110 OR diastolic > 80)
```

Time-series data: remote patient monitoring

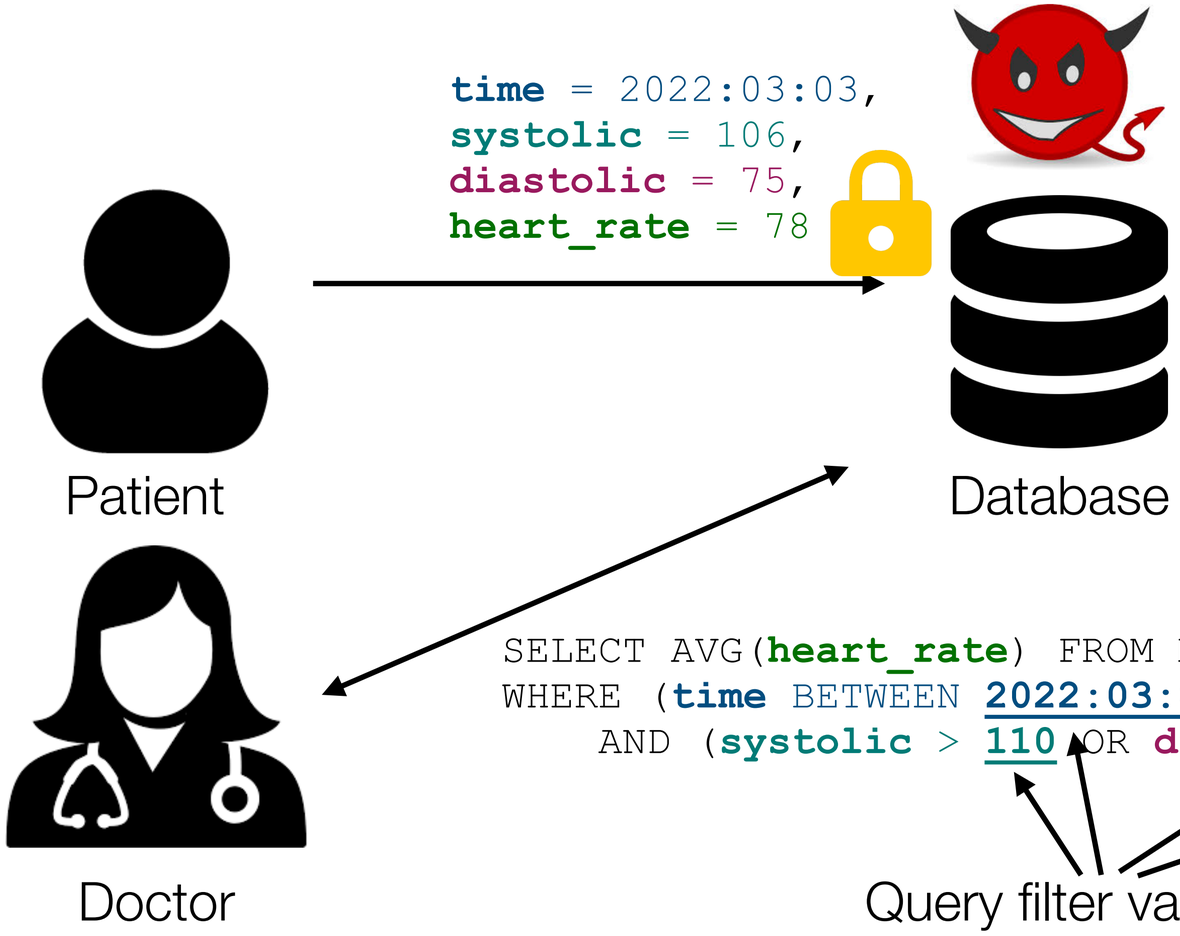


time	systolic	diastolic	heart_rate
2022:03:00	110	70	80
2022:03:01	112	68	72
2022:03:02	108	72	76

```
SELECT AVG(heart_rate) FROM MedicalHistory
WHERE (time BETWEEN 2022:03:01 and 2022:03:02)
    AND (systolic > 110 OR diastolic > 80)
```

Security: Protect data contents, query filter values

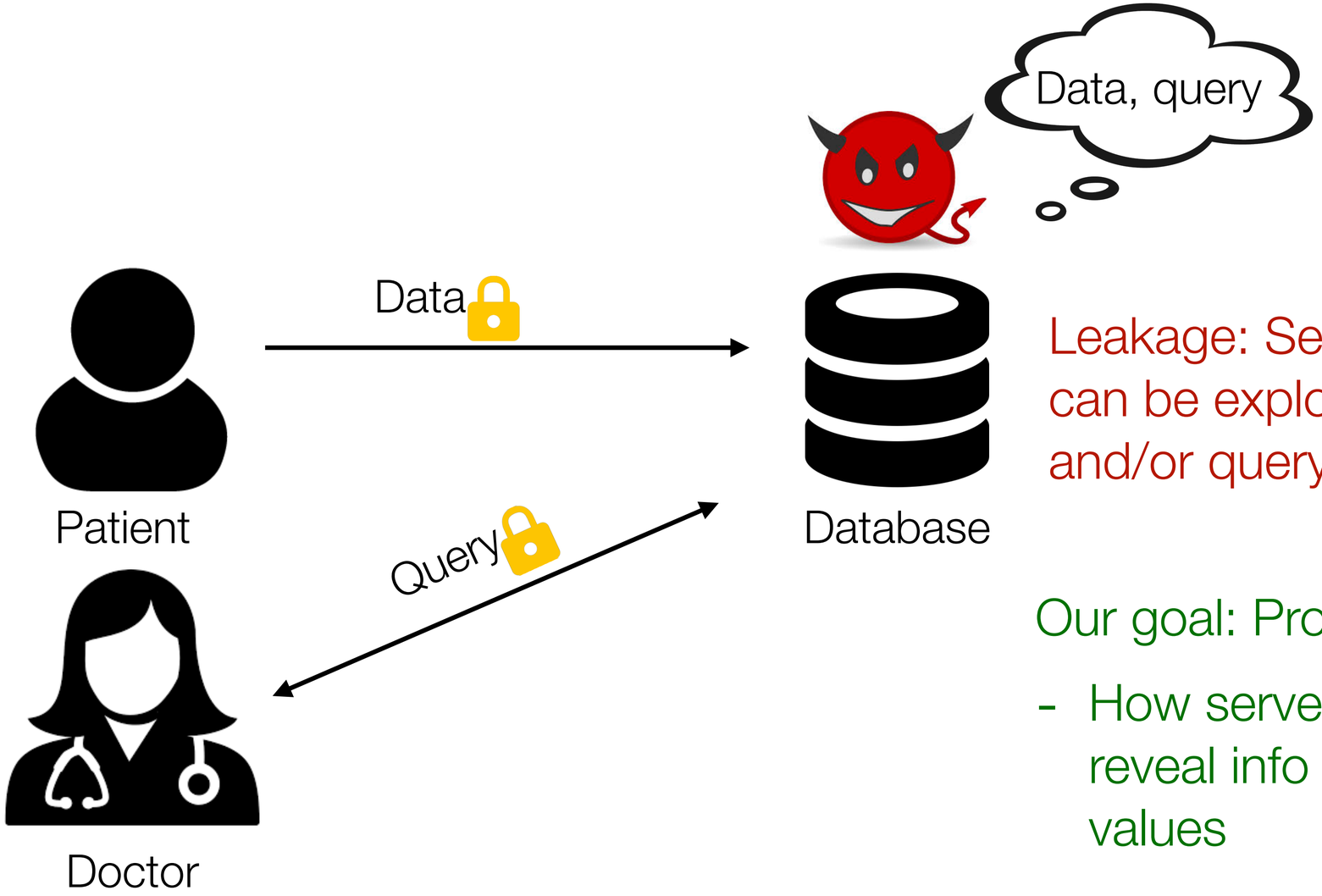
Time-series data: remote patient monitoring



time	systolic	diastolic	heart_rate
2022:03:00	110	70	80
2022:03:01	112	68	72
2022:03:02	108	72	76

Security: Protect data contents, query filter values

Prior work on encrypted databases [CryptDB, Arx, BlindSeer, …]



Leakage: Server learns information that can be exploited to recover the data and/or query [IKK12, KKKO16, ZKP16, GLMP19, …].

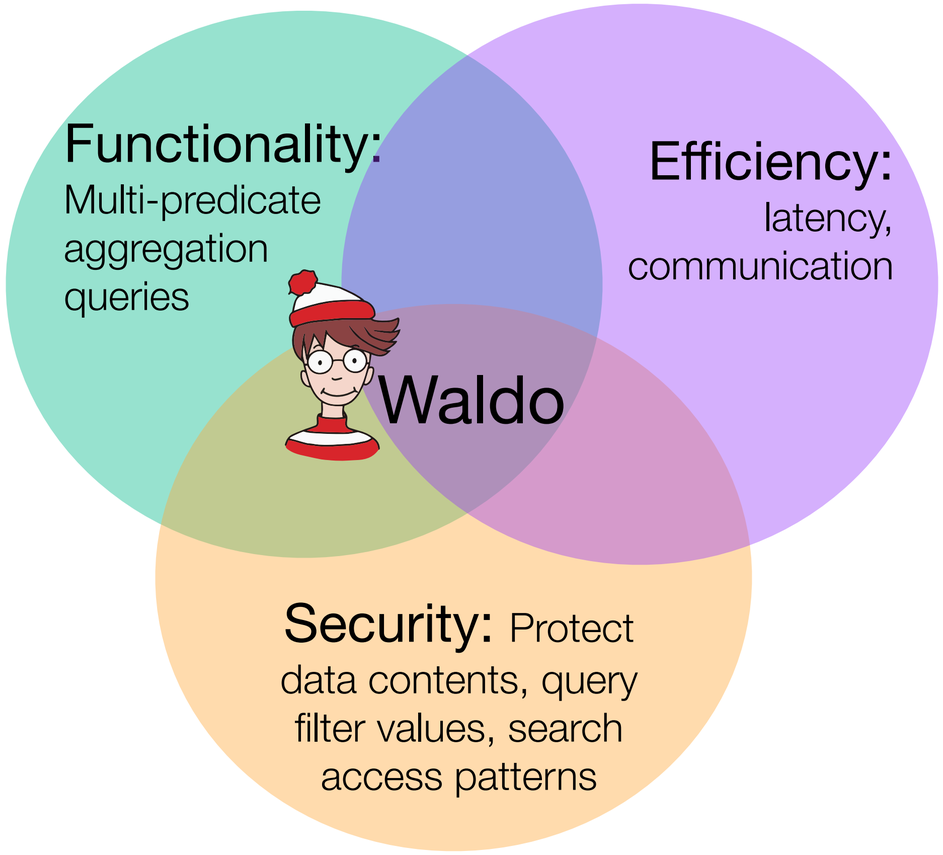
Our goal: Protect search access patterns

- How server accesses data should not reveal info about data or query filter values

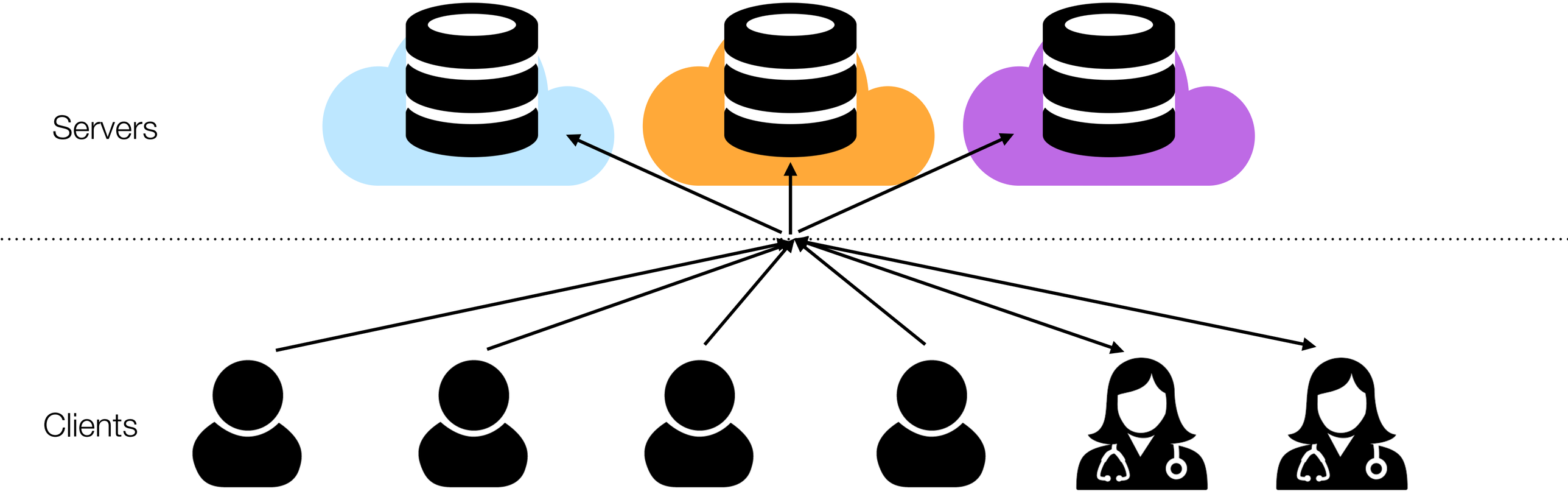
Finding a solution

Challenge: tools for hide search access patterns are expensive

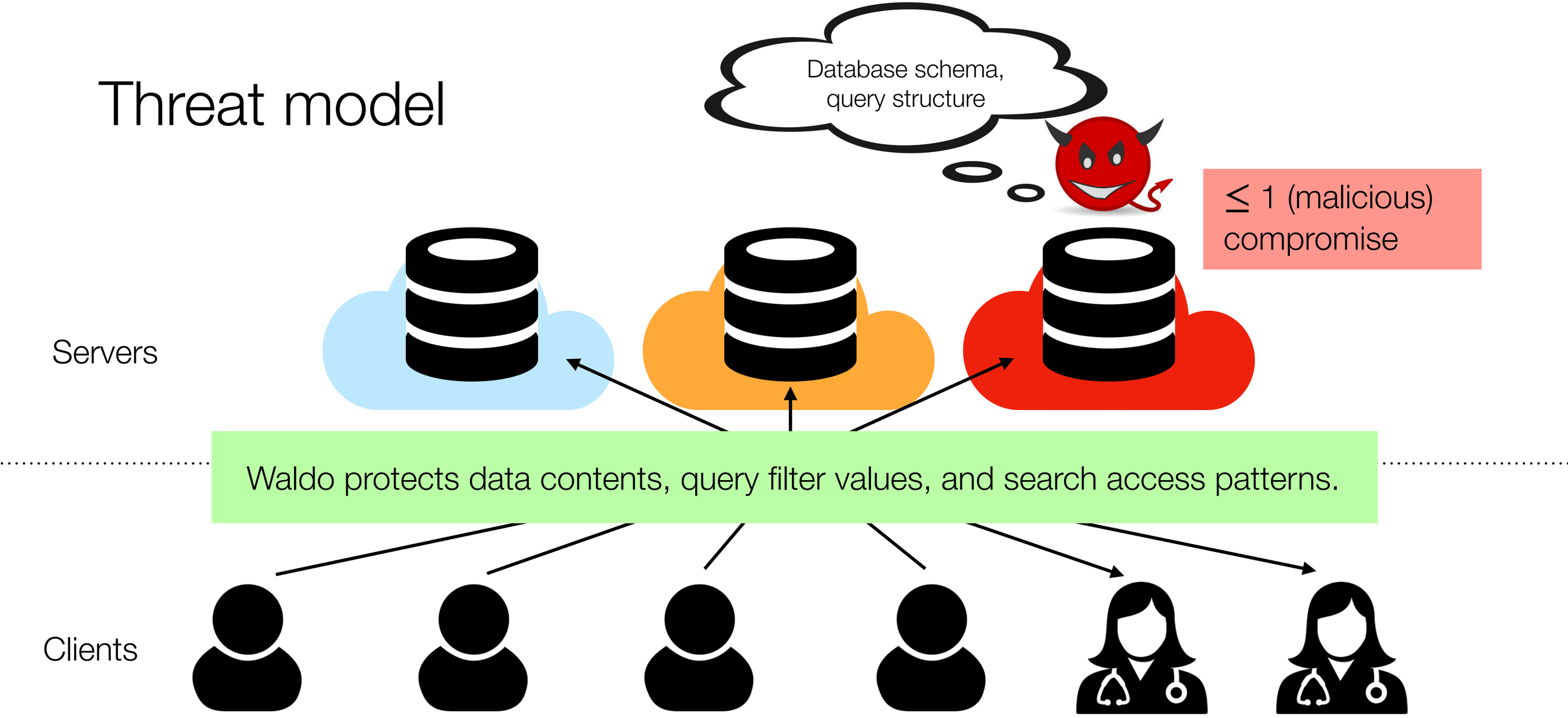
- e.g. ORAM [GO96, PathORAM], MPC [GMW87, BGW88]



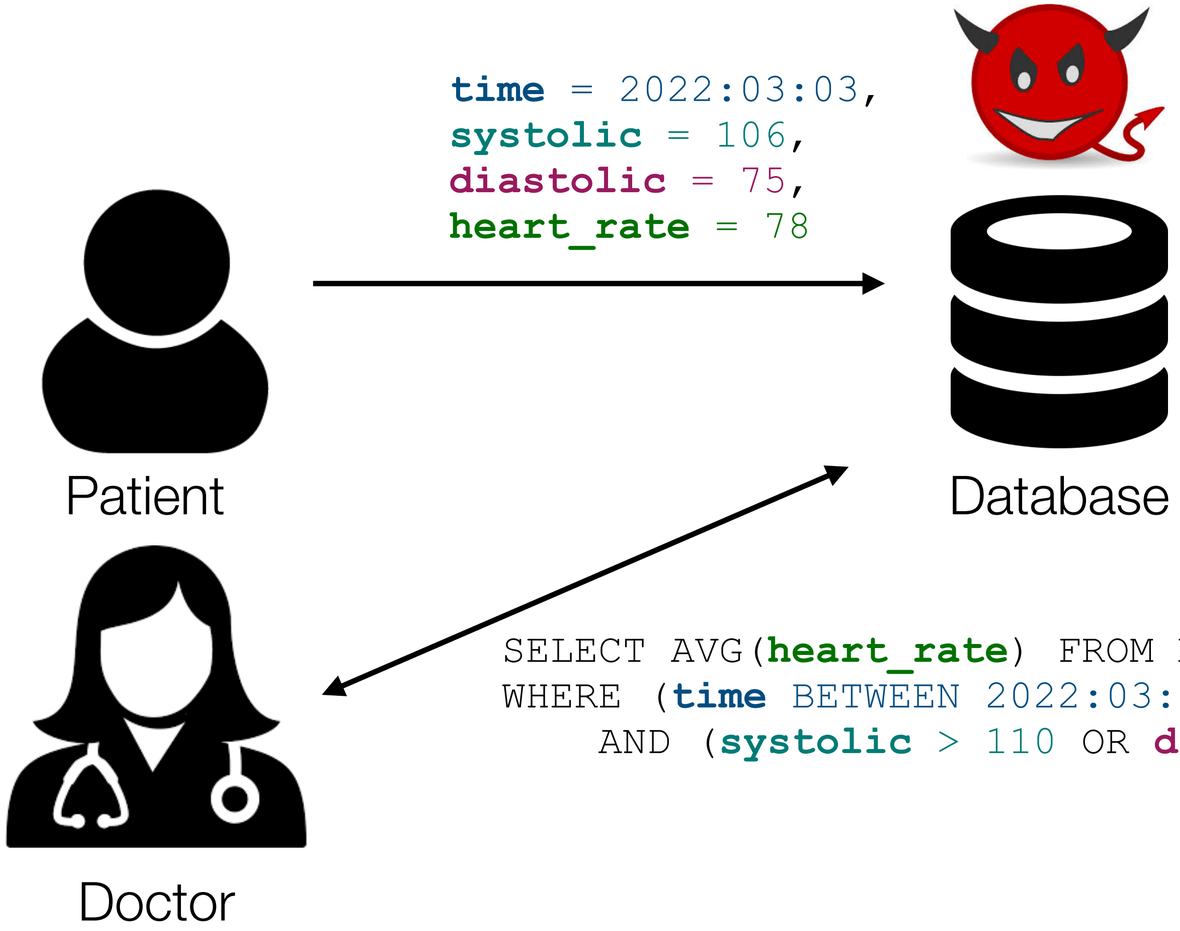
Threat model



Threat model

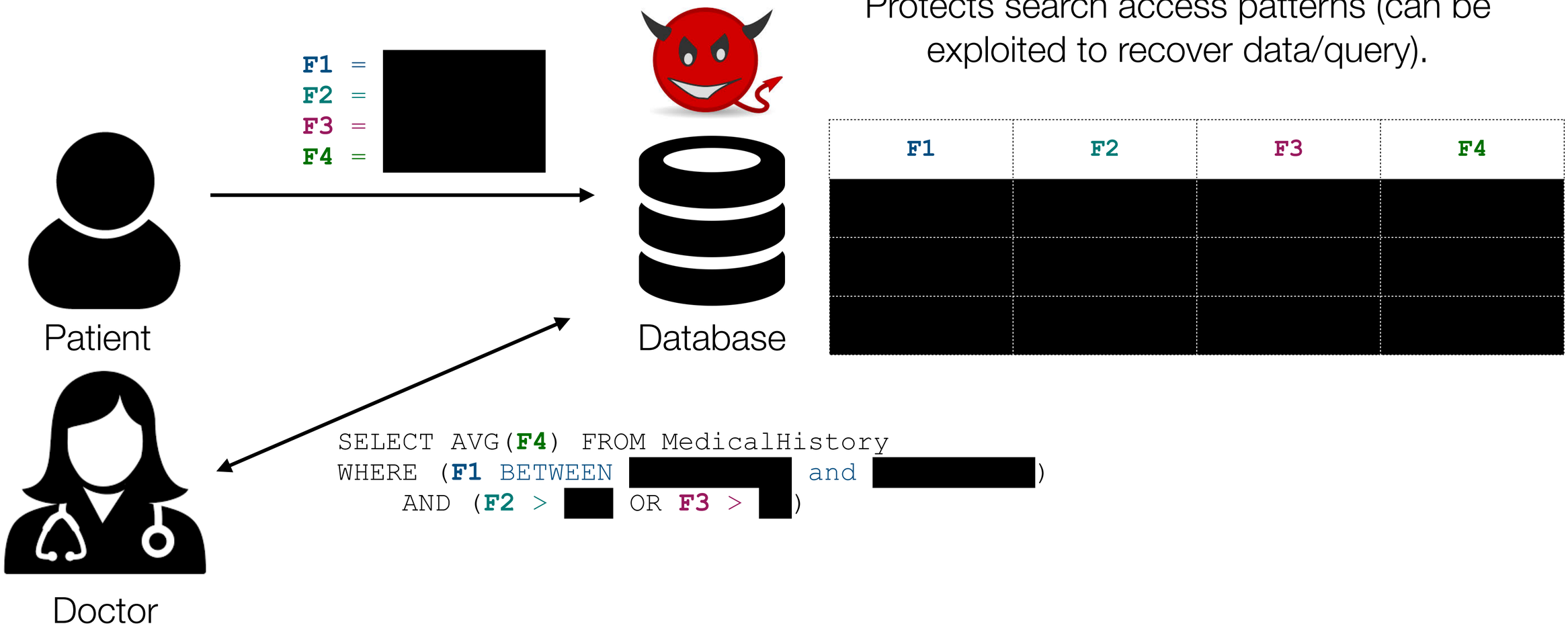


Waldo protects data contents, query filter values, and search access patterns



time	systolic	diastolic	heart_rate
2022:03:00	110	70	80
2022:03:01	112	68	72
2022:03:02	108	72	76

Waldo protects data contents, query filter values, and search access patterns



Protects search access patterns (can be exploited to recover data/query).

Notation

Split $x \in 2^\ell$ into secret shares $[x]_1, \ldots, [x]_n$ such that $\sum_{i=1}^{n} [x]_i = x$

Tool: Function Secret Sharing (FSS) [BGI15, BGI16, BGI19, BCG+21]

Two-party FSS scheme for function f splits function into 2 succinct shares ("keys").

Given a key, parties can compute shares of $f(x)$ without learning f.

Syntax:

- Gen$(f) \to K_1, K_2$: Generate keys for function f that hide f
- Eval$(K, x) \to y$: Given key and input value x, output $y = [f(x)]$

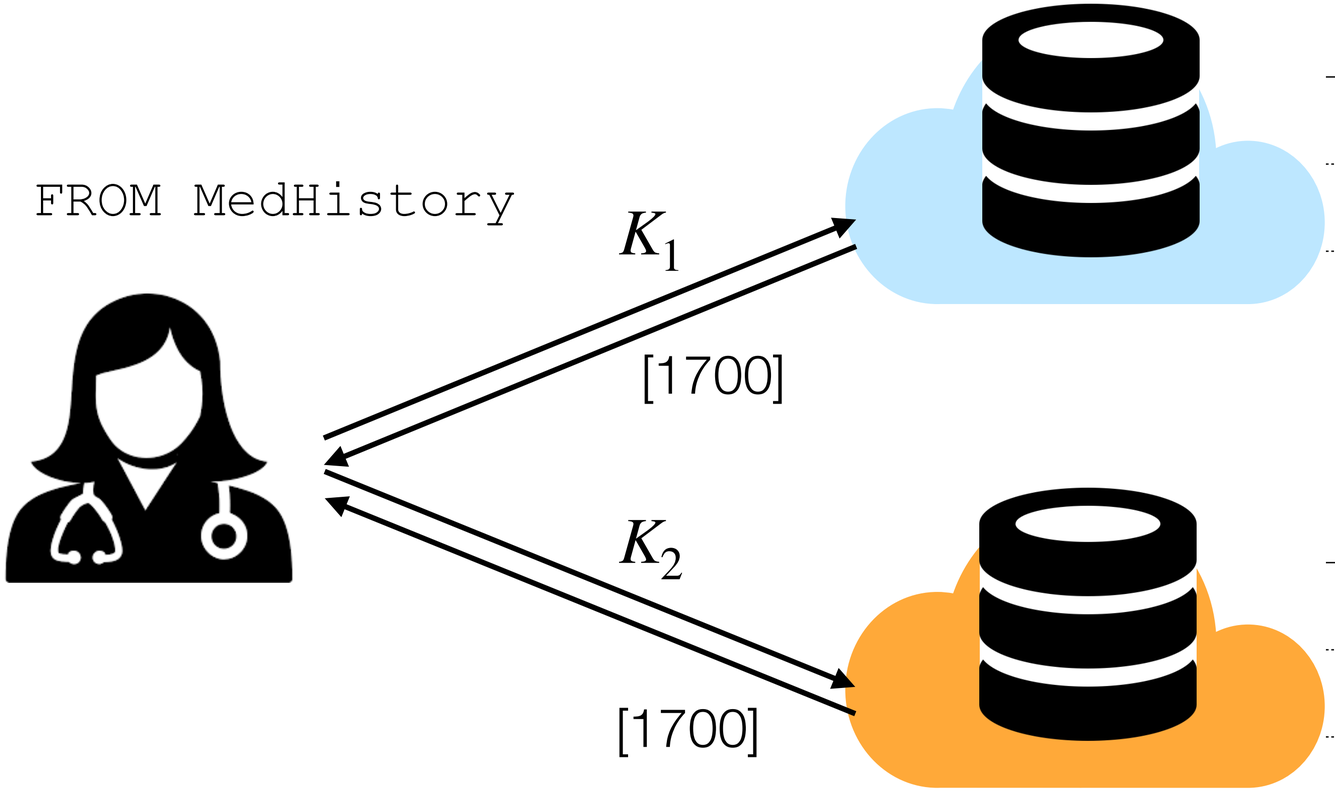
Efficient constructions exist for point functions and comparison functions.

Aggregate queries for public data

```
SELECT SUM(cost) FROM MedHistory
WHERE (ID = 112)
```

$K_1, K_2 \leftarrow \text{Gen}(f_{112 \to 1})$

Output 1700



ID	cost	
112	1000	$\times$ Eval(K_1,112) +
176	500	$\times$ Eval(K_1,176) +
112	700	$\times$ Eval(K_1,112)

ID	cost	
112	1000	$\times$ Eval(K_2,112) +
176	500	$\times$ Eval(K_2,176) +
112	700	$\times$ Eval(K_2,112)

Challenge: How do we extend these techniques to private data?

Idea: leverage search index structure

Entry $(i, j) = 1$ if record i has `heart_rate` j, 0 otherwise

Record #

heart_rate	1	2	3	...
...				
79	0	0	1	
80	1	1	0	
81	0	0	0	
...				

Record 3 has `heart_rate = 79`

Idea: leverage search index structure

Record #

heart_rate	1	2	3	4
...				
79	0	0	1	0
80	1	1	0	0

Appends are cheap → append to table

Target apps querying recent data

Limitation: Constrains domain of filtered features

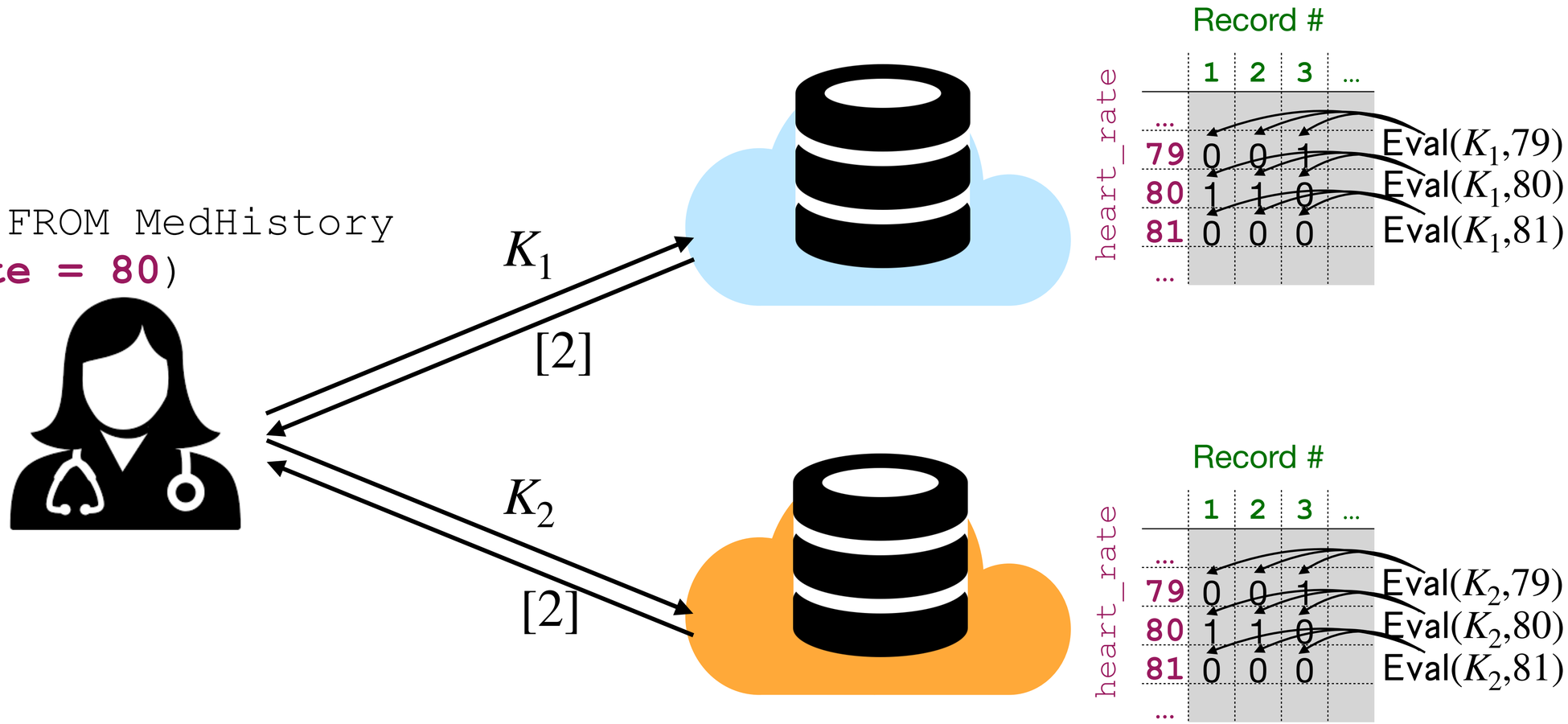
- In applications we examined, features in predicates already from small domain (e.g. 2^8) or can easily be mapped to one
  - EX: age, height, heart rate, blood glucose level, percentage
- Represent large domains via conjunctions
- Only restricts filtered features (not aggregated features).

Idea: leverage search index structure

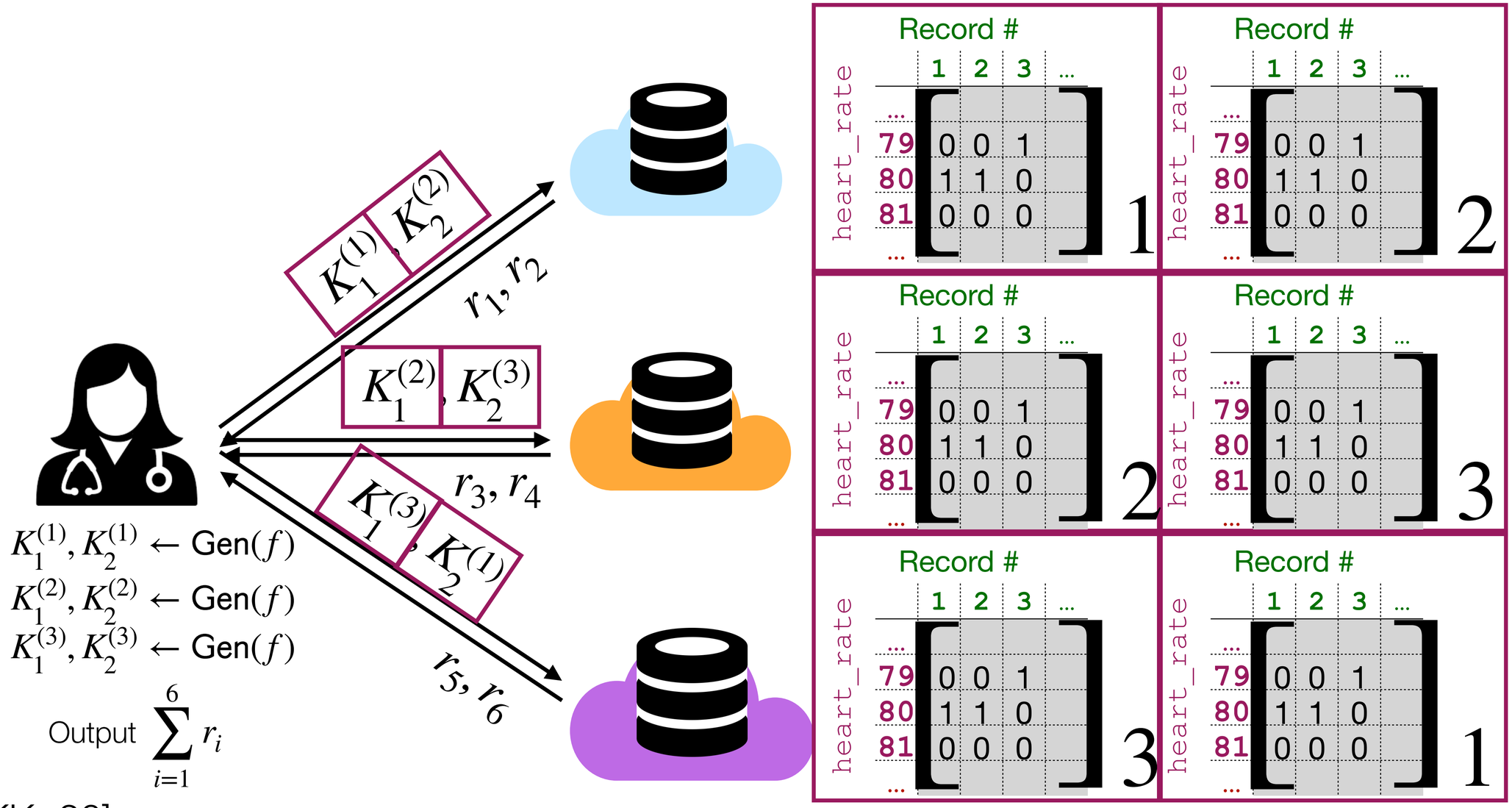
```
SELECT COUNT(*) FROM MedHistory
WHERE (heart_rate = 80)
```

$K_1, K_2 \leftarrow \text{Gen}(f_{80 \rightarrow 1})$

Output 2



Replicated secret sharing (RSS) with FSS



$K_1^{(1)}, K_2^{(1)} \leftarrow \text{Gen}(f)$

$K_1^{(2)}, K_2^{(2)} \leftarrow \text{Gen}(f)$

$K_1^{(3)}, K_2^{(3)} \leftarrow \text{Gen}(f)$

Output $\sum_{i=1}^{6} r_i$

[BKK+20]

Other contributions (see paper)

- Support for filtering based on multiple predicates
- Tools for cheap malicious security
- Construction for complex aggregates over time ranges
  - \+ Supports non-linear aggregation functions (e.g. min, max, top-k)
  - \- Only filters on time

Evaluation

Code available at: `https://github.com/ucbrise/waldo`

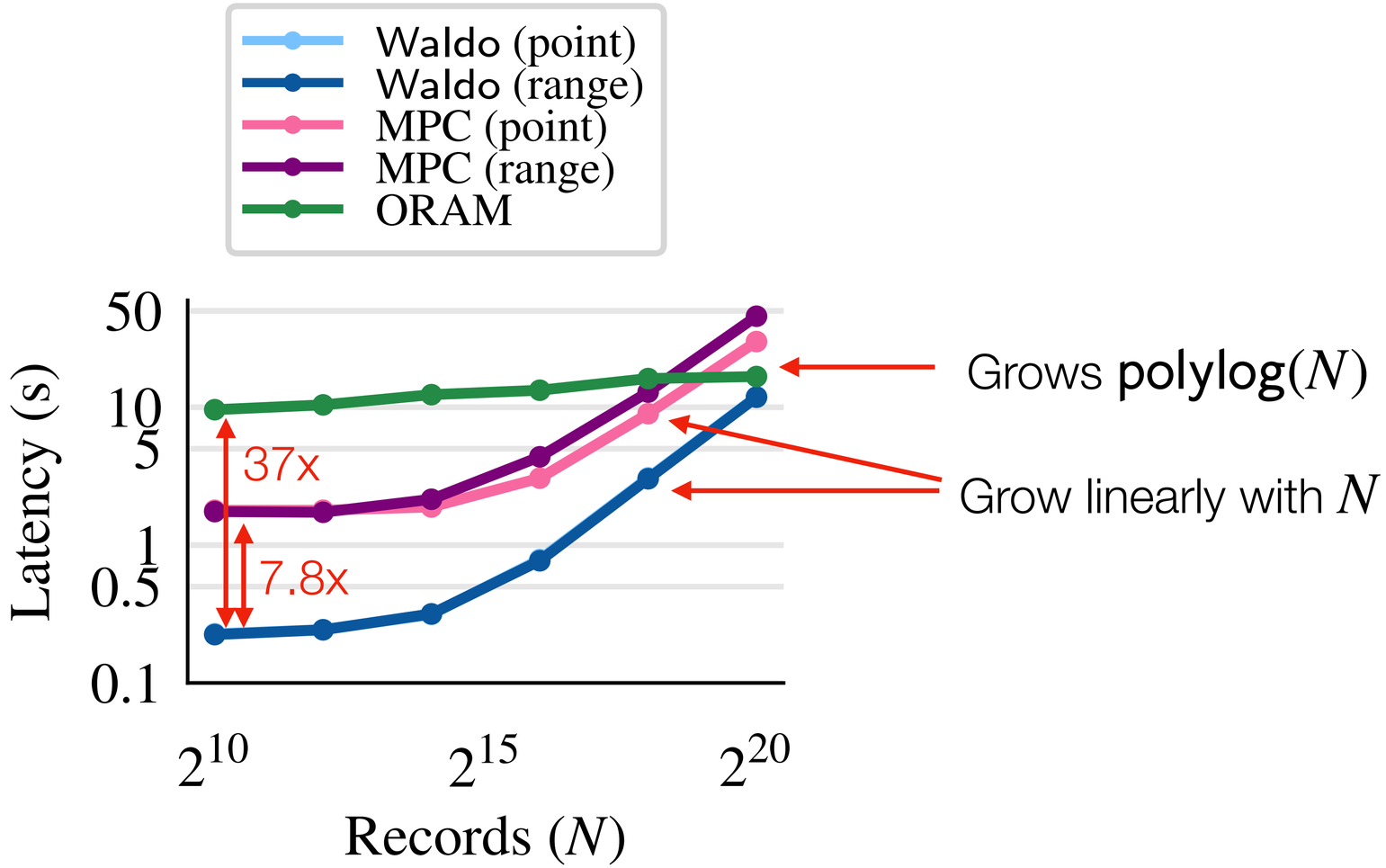
Experiment setup:

- Three 32-core servers
- 3 Gbps connection with 20ms RTT
- 8-predicate queries

Baselines (see paper):

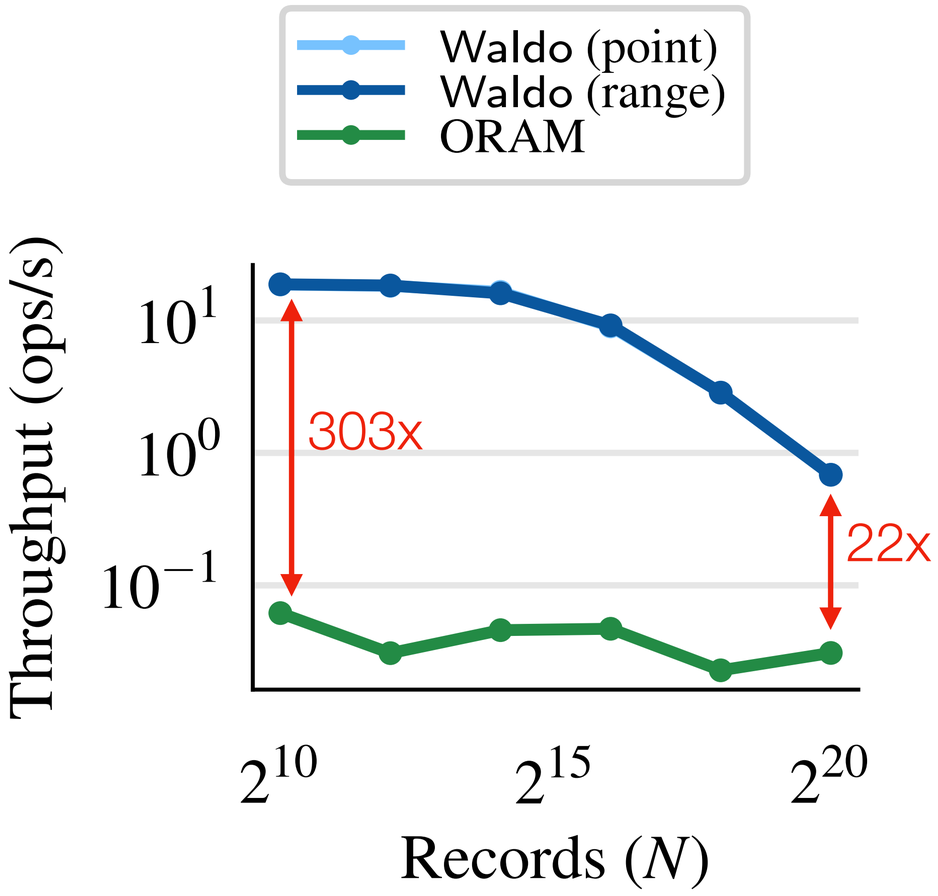
- Oblivious multidimensional tree (R-tree with PathORAM)
- Generic MPC: MP-SPDZ (3PC honest-majority)

Latency



Point and range almost identical

Throughput


Waldo (point)
Waldo (range)
ORAM
Throughput (ops/s)
10^1
10^0
10^{-1}
303x
22x
2^{10}
2^{15}
2^{20}
Records (N)

Conclusion

Waldo is a time-series database that provides:

- *Functionality*: Supports multi-predicate queries
- *Security:* Hides data contents, query filter values, and search access patterns
- *Efficiency*: Low latency, high throughput, low bandwidth

Thanks!

Emma Dauterman
edauterman@berkeley.edu

https://eprint.iacr.org/2021/1661.pdf
https://github.com/ucbrise/waldo