CS 599: Formal Methods in Security and Privacy

Differential Privacy

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

Where we were…

(ε,δ)-Differential Privacy

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query $Q: X^n \rightarrow R$ is (ε,δ)-differentially private iff

for all adjacent database b_1, b_2 and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon)\Pr[Q(b_2) \in S] + \delta$$

apRHL

Indistinguishability parameter

Precondition (a logical formula)

$$\vdash_{\epsilon,\delta} c_1 \sim c_2 : P \Rightarrow Q$$

Probabilistic Program

Probabilistic Program

Postcondition (a logical formula)

apRHL: More general Lap rule (still restricted)

$$\vdash_{k*\varepsilon,0} \frac{}{\begin{array}{l}\texttt{x}_1 := \$ \ \texttt{Lap}(1/\varepsilon, \texttt{y}_1) \\ \sim \\ \texttt{x}_2 := \$ \ \texttt{Lap}(1/\varepsilon, \texttt{y}_2) \\ : \ |\texttt{y}_1 - \texttt{y}_2| \le k \Rightarrow =\end{array}}$$

Probabilistic Relational Hoare Logic
Composition

$$\frac{\vdash_{\varepsilon 1,\delta 1} c_1 \sim c_2 : P \Rightarrow R \qquad \vdash_{\varepsilon 2,\delta 2} c_1' \sim c_2' : R \Rightarrow S}{\vdash_{\varepsilon 1+\varepsilon 2,\delta 1+\delta 2} c_1 ; c_1' \sim c_2 ; c_2' : P \Rightarrow S}$$

apRHL
awhile

$$\frac{\begin{array}{c} P \land e<1> \le 0 \Rightarrow \neg b1<1> \\[1em] \vdash_{\epsilon_k,\delta_k} c1 \sim c2 : P \land b1<1> \land b2<2> \land k=e<1> \land e<1> \le n \\ \Longrightarrow P \land b1<1> = b2<2> \land k < e<1> \end{array}}{\begin{array}{c} \text{while } b1 \text{ do } c1 \sim \text{while } b2 \text{ do } c2 \\ \vdash_{\sum\epsilon_k,\sum\delta_k} : P \land b1<1> = b2<2> \land e<1> \le n \\ \Longrightarrow P \land \neg b1<1> \land \neg b2<2> \end{array}}$$

Releasing partial sums

```
DummySum(d : {0,1} list) : real list
   i:= 0;
   s:= 0;
   r:= [];
   while (i<size d)
      s:= s + d[i]
      z:=$ Lap(eps,s)
      r:= r ++ [z];
      i:= i+1;
   return r
```

I am using the easycrypt notation here where `Lap(eps,a)` corresponds to adding to the value `a` noise from the Laplace distribution with `b=1/eps` and mean `mu=0`.

Releasing partial sums

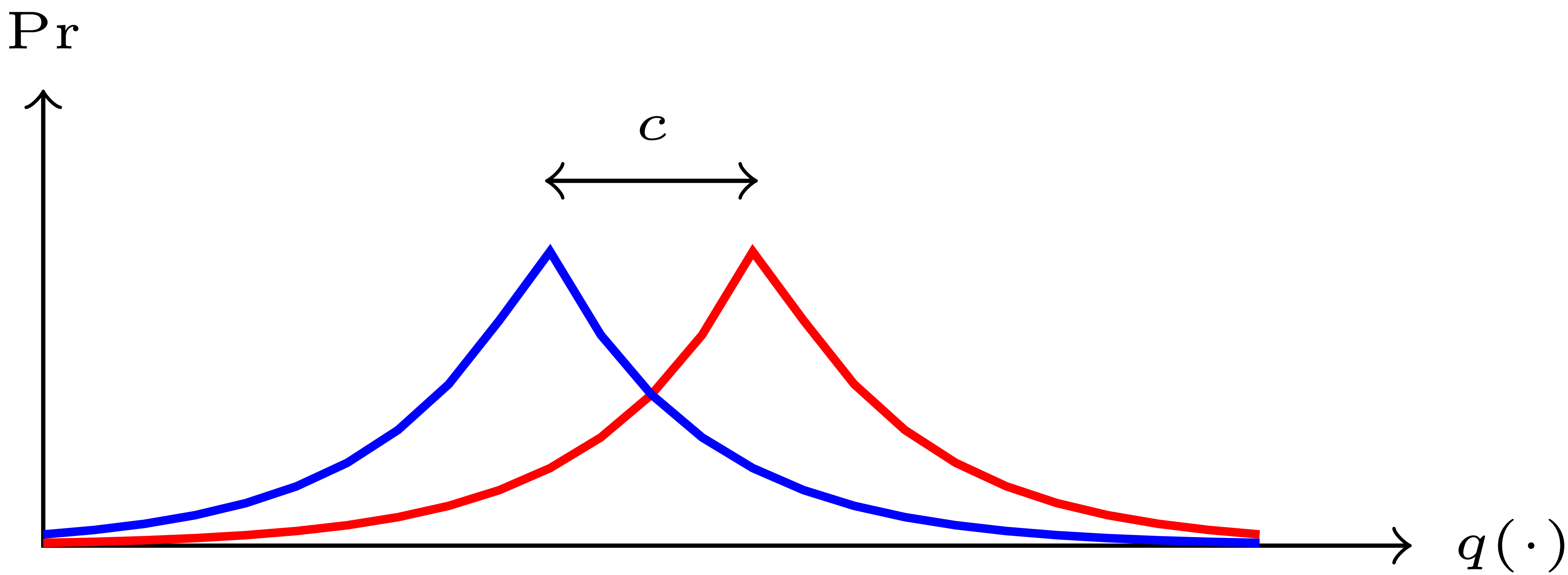
```
DummySum(d : {0,1} list) : real list
   i:=0;
   s:=0;
   r:=[];
   while (i<size d)
      z:=$ Lap(eps,d[i])
      s:= s + z
      r:= r ++ [s];
      i:= i+1;
   return r
```

Today: more examples of differentially private programs

Laplace Mechanism

Theorem (Privacy of the Laplace Mechanism)
The Laplace mechanism is ε-differentially private.

Proof: Intuitively



Exponential Mechanism

The Exponential Mechanism can be used in more situations - accordingly to a score function.

Suppose that we have a scoring function u(D,o) that to each pair (database, potential output) assign a score (a negative real number).

We want to output approximately the element with the max score.

Exponential Mechanism

Exponential Mechanism:

$\mathcal{M}_E(x, u, \mathcal{R})$

return $r \in \mathcal{R}$ with prob. $\dfrac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}$

where

$$\Delta u = \max_{r \in \mathcal{R}} \max_{x \sim_1 y} \left| u(x, r) - u(y, r) \right|$$

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

The proof is very similar to the one for the Laplace Mechanism.

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

The proof is very similar to the one for the Laplace Mechanism.

$$\frac{\Pr[\mathcal{M}_E(x,u,\mathcal{R})=r]}{\Pr[\mathcal{M}_E(y,u,\mathcal{R})=r]} = \frac{\left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)}{\left(\frac{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(y,r')}{2\Delta u})}\right)}$$

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

The proof is very similar to the one for the Laplace Mechanism.

$$\frac{\Pr[\mathcal{M}_E(x,u,\mathcal{R})=r]}{\Pr[\mathcal{M}_E(y,u,\mathcal{R})=r]} = \frac{\left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)}{\left(\frac{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(y,r')}{2\Delta u})}\right)}$$

$$= \left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}\right)\cdot\left(\frac{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

The proof is very similar to the one for the Laplace Mechanism.

$$\frac{\Pr[\mathcal{M}_E(x,u,\mathcal{R})=r]}{\Pr[\mathcal{M}_E(y,u,\mathcal{R})=r]} = \frac{\left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)}{\left(\frac{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(y,r')}{2\Delta u})}\right)}$$

$$= \left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}\right)\cdot\left(\frac{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

$$= \exp\left(\frac{\varepsilon(u(x,r')-u(y,r'))}{2\Delta u}\right)\cdot\left(\frac{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r'\in\mathcal{R}}\exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

Continuing

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

Continuing

$$= \exp\left(\frac{\varepsilon(u(x,r') - u(y,r'))}{2\Delta u}\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

Continuing

$$= \exp\left(\frac{\varepsilon(u(x,r') - u(y,r'))}{2\Delta u}\right) \cdot \left(\frac{\sum_{r'\in\mathcal{R}} \exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r'\in\mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

$$\leq \exp\left(\frac{\varepsilon}{2}\right) \cdot \exp\left(\frac{\varepsilon}{2}\right) \cdot \left(\frac{\sum_{r'\in\mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}{\sum_{r'\in\mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

Exponential Mechanism

Privacy theorem:
The Exponential Mechanism is differentially private.

Continuing

$$= \exp\left(\frac{\varepsilon(u(x,r') - u(y,r'))}{2\Delta u}\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

$$\leq \exp\left(\frac{\varepsilon}{2}\right) \cdot \exp\left(\frac{\varepsilon}{2}\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)$$

Here we change y with x by paying exp(ε/2).

Exponential Mechanism

The Exponential Mechanism is a very general mechanism. It can actually be used as a kind of universal mechanism.

Unfortunately, when the output space is big it can be very costly to sample from it - the best option is to enumerate all the possibilities.

Moreover, when the output space is big also the accuracy get worse.

Report Noisy Max

(a)

(b)

(c)

(d)

Suppose that each one of us can vote for one star, and we want to say who is the star that receives most votes.

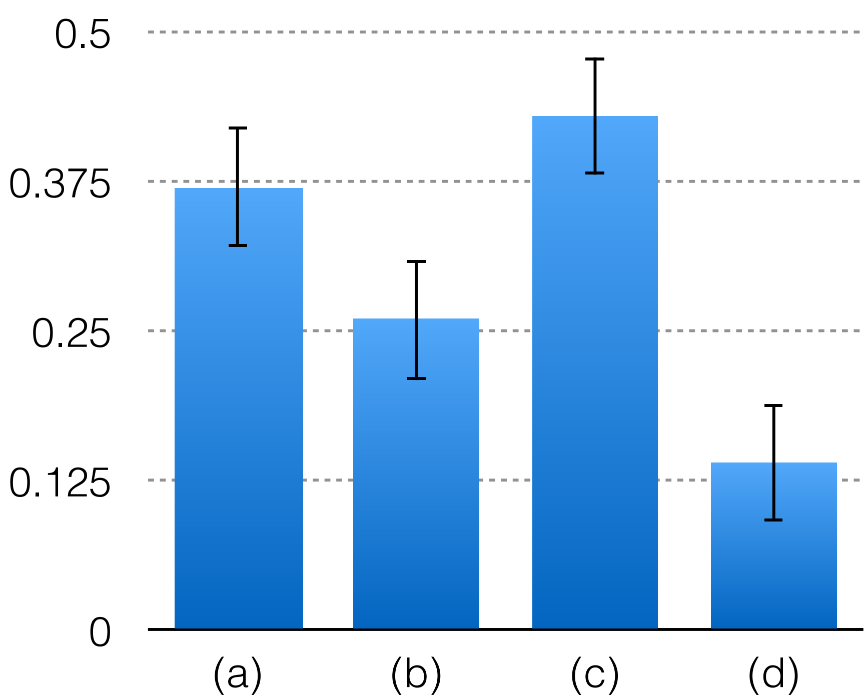
Report Noisy Max

(a)

(b)

(c)

(d)



Intuition:
We can compute the histogram add Laplace noise to each score and then select the maximal noised score.

Report Noisy Max

Given a set of queries with sensitivity 1, return the index of the noised query with the max value

$q_1(D)$+noise

$q_2(D)$+noise

$q_3(D)$+noise

.....

$q_k(D)$+noise

Report Noisy Max

Given a set of queries with sensitivity 1, return the index of the noised query with the max value

$q_1(D)$+noise

$q_2(D)$+noise

$q_3(D)$+noise

.....

$q_k(D)$+noise

A naive analysis gives kε-differentially private

Report Noisy Max - intuition

We can prove this algorithm
ε-differentially private

Report Noisy Max - intuition

We can prove this algorithm
ε-differentially private

Databases differing
in one individual

Report Noisy Max - intuition

$q_1(D)$+noise	$q_1(D')$+noise
$q_2(D)$+noise	$q_2(D')$+noise
$q_3(D)$+noise	$q_3(D')$+noise
.....	.....
$q_k(D)$+noise	$q_k(D')$+noise

Databases differing in one individual

We can prove this algorithm ε-differentially private

Report Noisy Max - intuition

1 sensitive queries

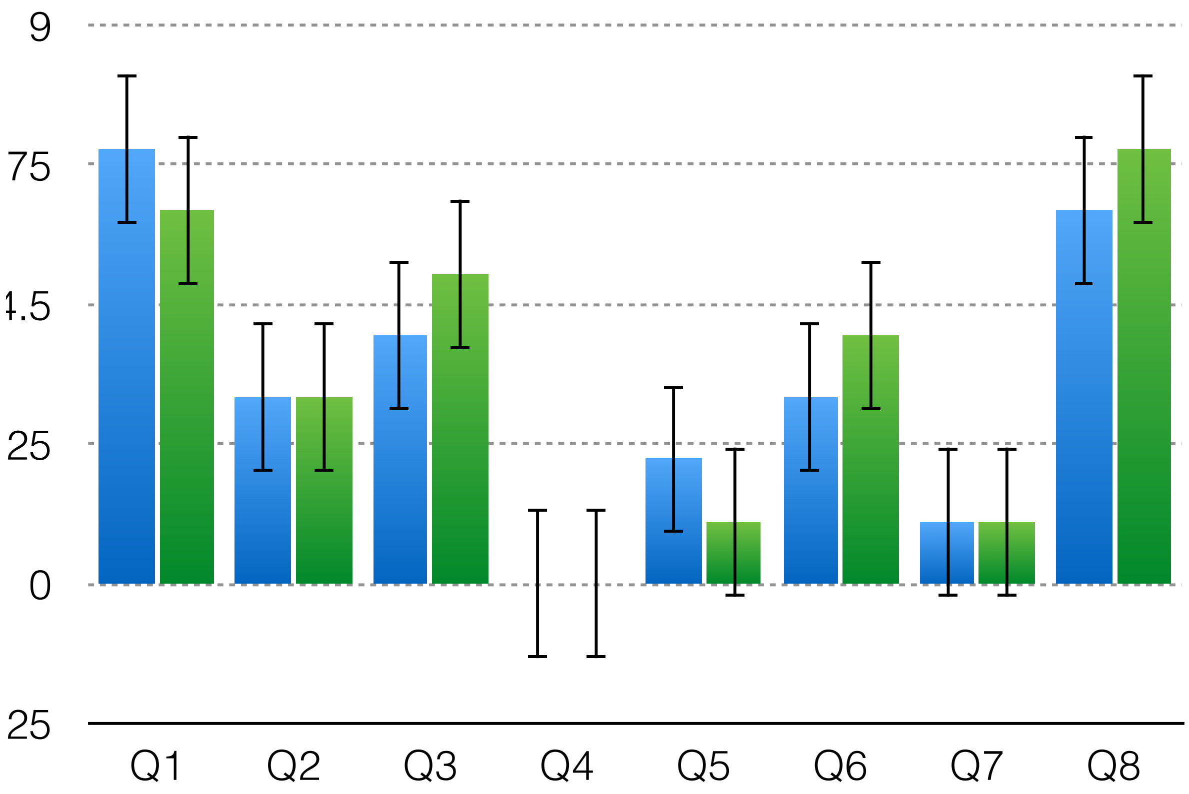
$q_1(D)$+noise	$q_1(D')$+noise
$q_2(D)$+noise	$q_2(D')$+noise
$q_3(D)$+noise	$q_3(D')$+noise
.....	.....
$q_k(D)$+noise	$q_k(D')$+noise

Databases differing in one individual

We can prove this algorithm ε-differentially private

Report Noisy Max - intuition

1 sensitive queries



$q_1(D)$+noise $q_1(D')$+noise

$q_2(D)$+noise $q_2(D')$+noise

$q_3(D)$+noise $q_3(D')$+noise

....

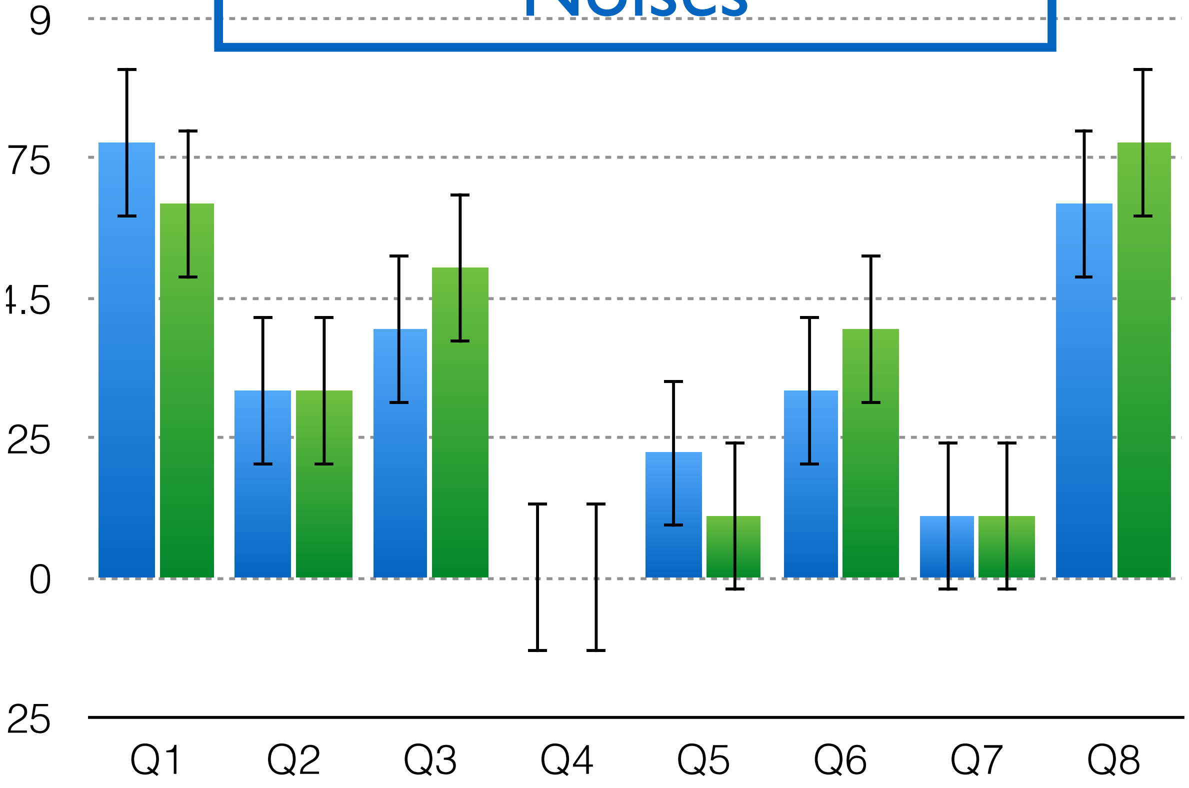
$q_k(D)$+noise $q_k(D')$+noise

Databases differing
in one individual

We can prove this algorithm
ε-differentially private

Report Noisy Max - intuition

We need to coordinate Noises



We can prove this algorithm ε-differentially private

1 sensitive queries

$q_1(D)$+noise	$q_1(D')$+noise
$q_2(D)$+noise	$q_2(D')$+noise
$q_3(D)$+noise	$q_3(D')$+noise
.....	.....
$q_k(D)$+noise	$q_k(D')$+noise

Databases differing in one individual

Report Noisy Max

```
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
     cur = qi(b) + Lap(1/ε)
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
```

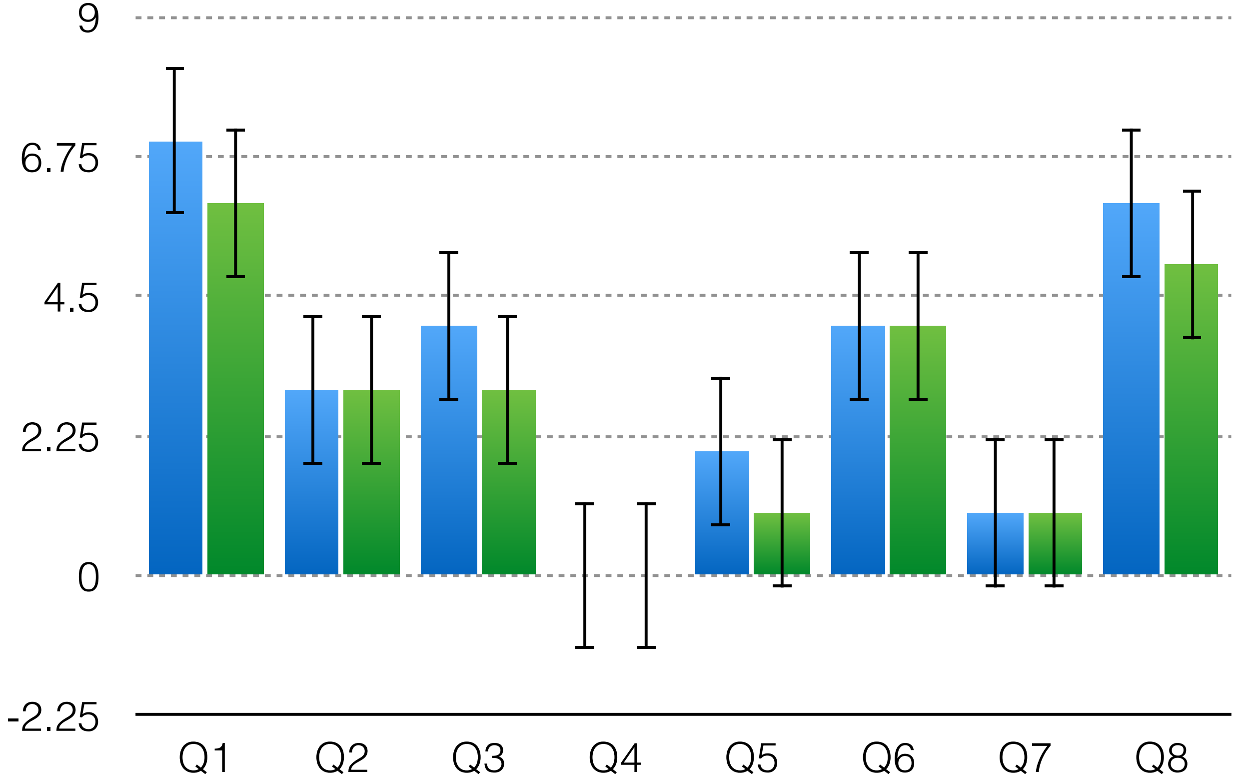
Report Noisy Max

Simplifying assumptions

$\mathtt{c}_{\mathtt{k}} \geq \mathtt{c}_{\mathtt{k}}'$

$\mathtt{c}_{\mathtt{k}}' + 1 \geq \mathtt{c}_{\mathtt{k}}$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Report Noisy Max

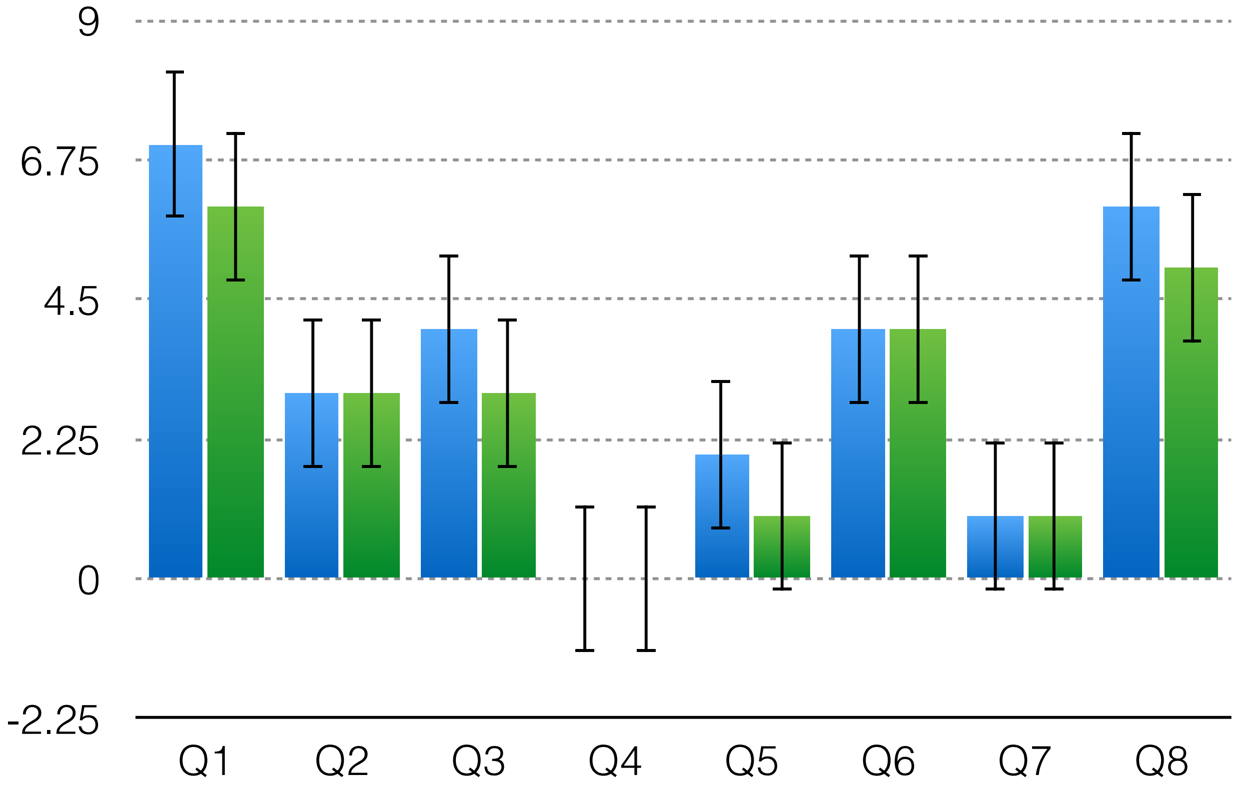
Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Report Noisy Max

Simplifying assumptions

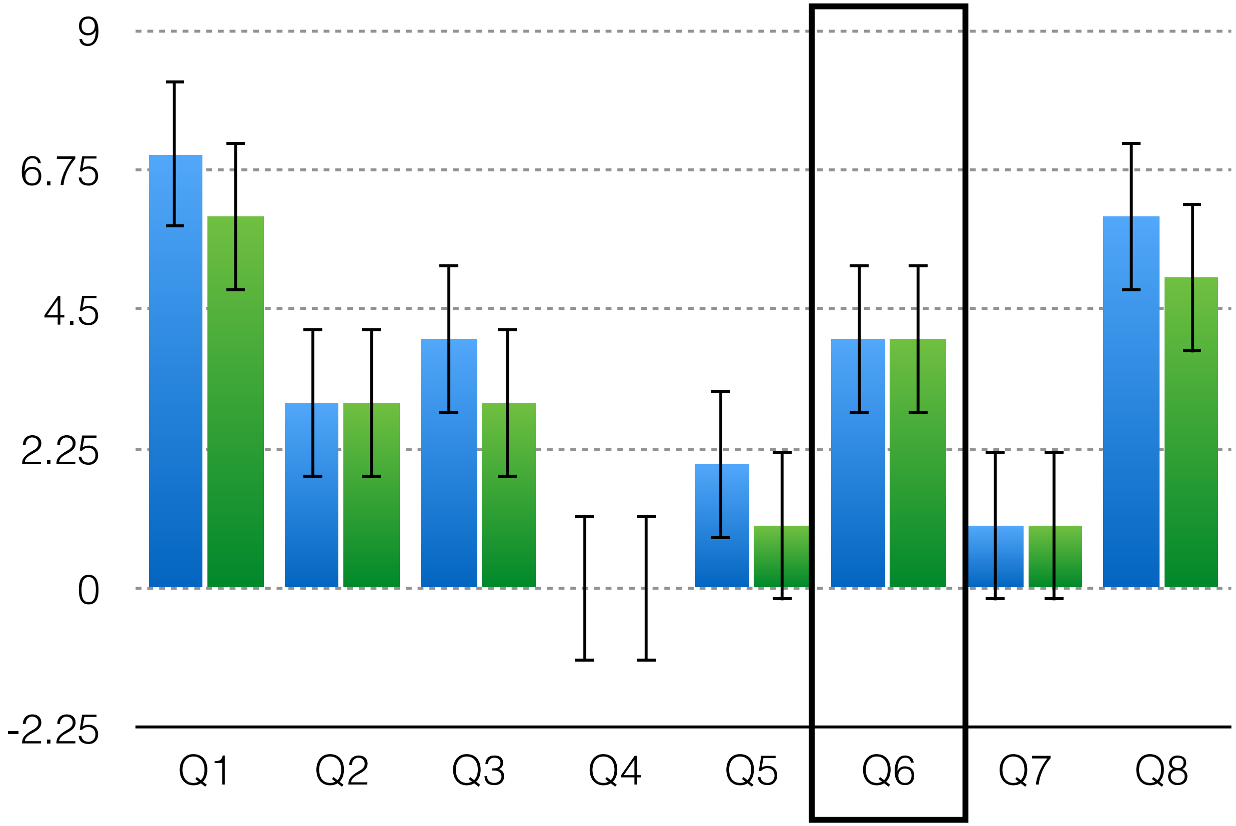
$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$
$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

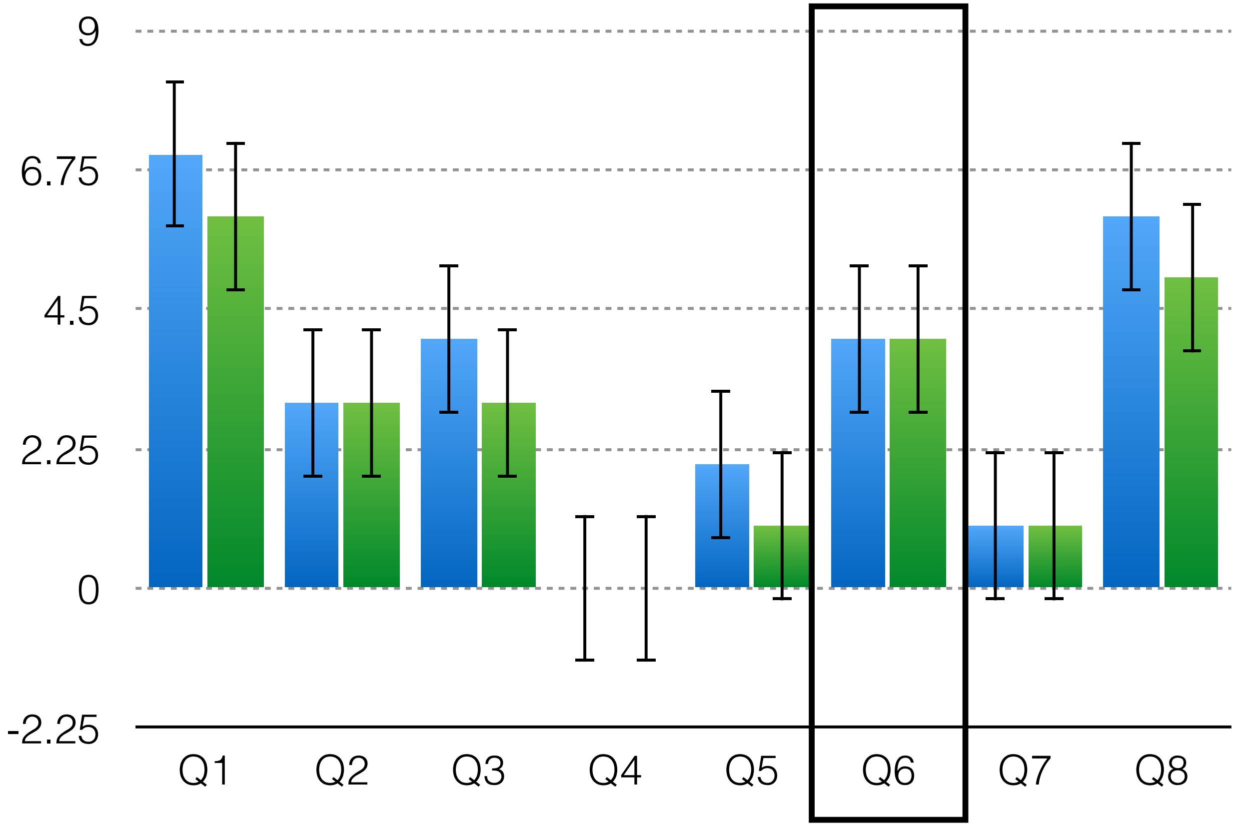
Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

We want to show:

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] \leq e^{\epsilon} \Pr_{x \sim RNM(D')}[x = i \mid r_{-i}]$$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

We want to show:

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] \leq e^{\epsilon} \Pr_{x \sim RNM(D')}[x = i \mid r_{-i}]$$

Report Noisy Max

Simplifying assumptions

$c_k \geq c_k'$

$c_k' + 1 \geq c_k$

Notation r_k, r_k'

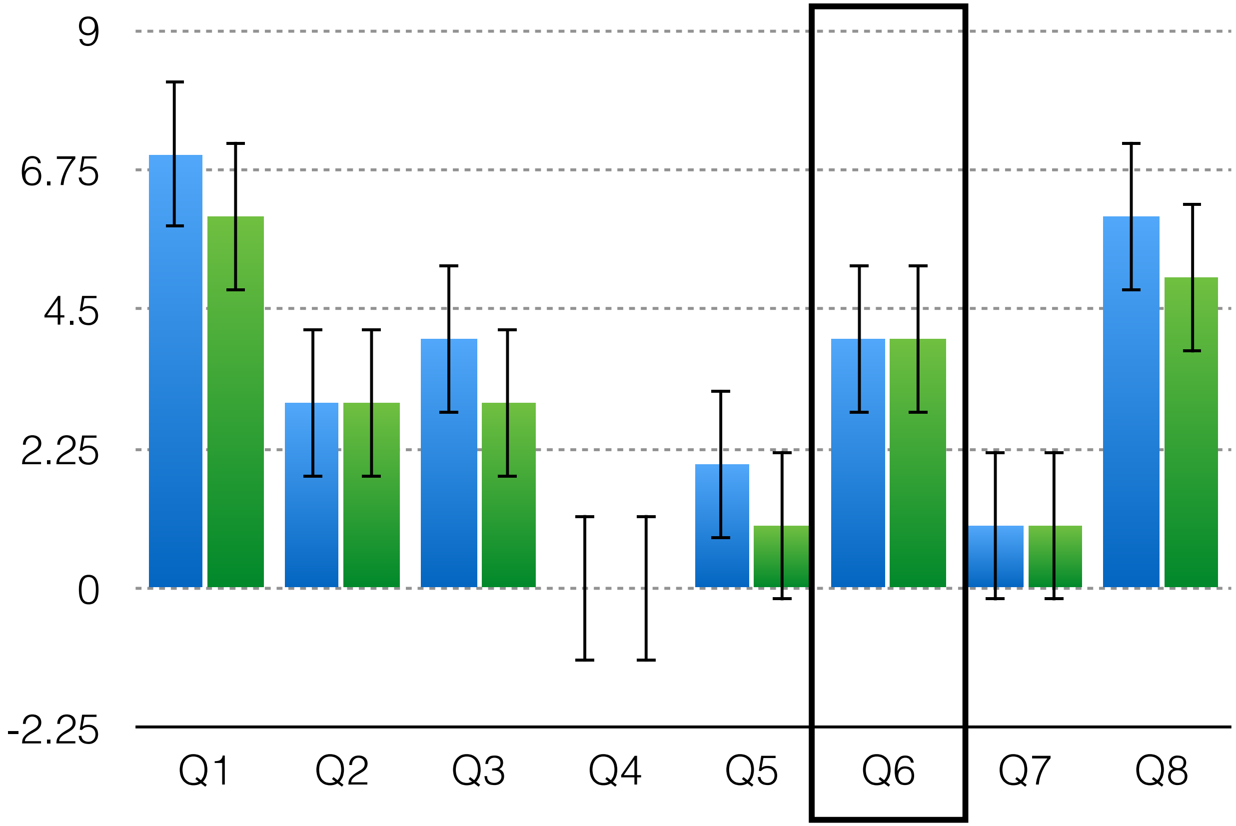
noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

By fixing the noises r_j for all $j \neq i$ we can compute the following

$\boxed{r^* = \min_r \; c_i + r \;\geq\; c_j + r_j \text{ for all } j}$

Report Noisy Max



Simplifying assumptions

$c_k \geq c_k'$
$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

By fixing the noises r_j for all $j \neq i$ we can compute the following

$\boxed{r^* = \min_r \; c_i + r \;\geq\; c_j + r_j \text{ for all } j}$

Report Noisy Max

$\boxed{r^* = \min_r \; c_i + r \geq c_j + r_j \text{ for all } j}$

Notice that

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] = \Pr_{r \sim Lap}[r \geq r^*]$$

Simplifying assumptions

$c_k \geq c_k'$
$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$\boxed{r^{*}=\min_r \; c_i+r \geq c_j+r_j \text{ for all } j}$

Notice that we also have

$$c_i'+1+r^{*} \geq c_j'+r_j$$

Since

$$c_i+r^{*} \geq c_j+r_j$$

and this says

$$\Pr_{x\sim RNM(D')}[x=i \mid r_{-i}] \geq \Pr_{r\sim Lap}[r \geq 1+r^{*}]$$

Simplifying assumptions

$c_k \geq c_k'$
$c_k'+1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$\boxed{r^* = \min_r \; c_i + r \geq c_j + r_j \text{ for all } j}$

Summarizing we have:

$$\Pr_{x \sim RNM(D)}[x = i \,|\, r_{-i}] = \Pr_{r \sim Lap}[r \geq r^*]$$

And

$$\Pr_{x \sim RNM(D')}[x = i \,|\, r_{-i}] \geq \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

Simplifying assumptions

$c_k \geq c_k'$
$c_k' + 1 \geq c_k$

Notation r_k, r_k' noise added at round k.

Let's focus on the iteration i and let's fix the noises r_j for all $j \neq i$

Report Noisy Max

$\boxed{\texttt{r*=min}_{\texttt{r}}\ \texttt{c}_{\texttt{i}}\texttt{+r} \geq \texttt{c}_{\texttt{j}}\texttt{+r}_{\texttt{j}} \text{ for all } \texttt{j}}$

Summarizing we have:

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}] = \Pr_{r \sim Lap}[r \geq r^*]$$

And

$$\Pr_{x \sim RNM(D')}[x = i \mid r_{-i}] \geq \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

How can we connect them?

Simplifying assumptions

$\texttt{c}_{\texttt{k}} \geq \texttt{c}_{\texttt{k}}'$

$\texttt{c}_{\texttt{k}}' + 1 \geq \texttt{c}_{\texttt{k}}$

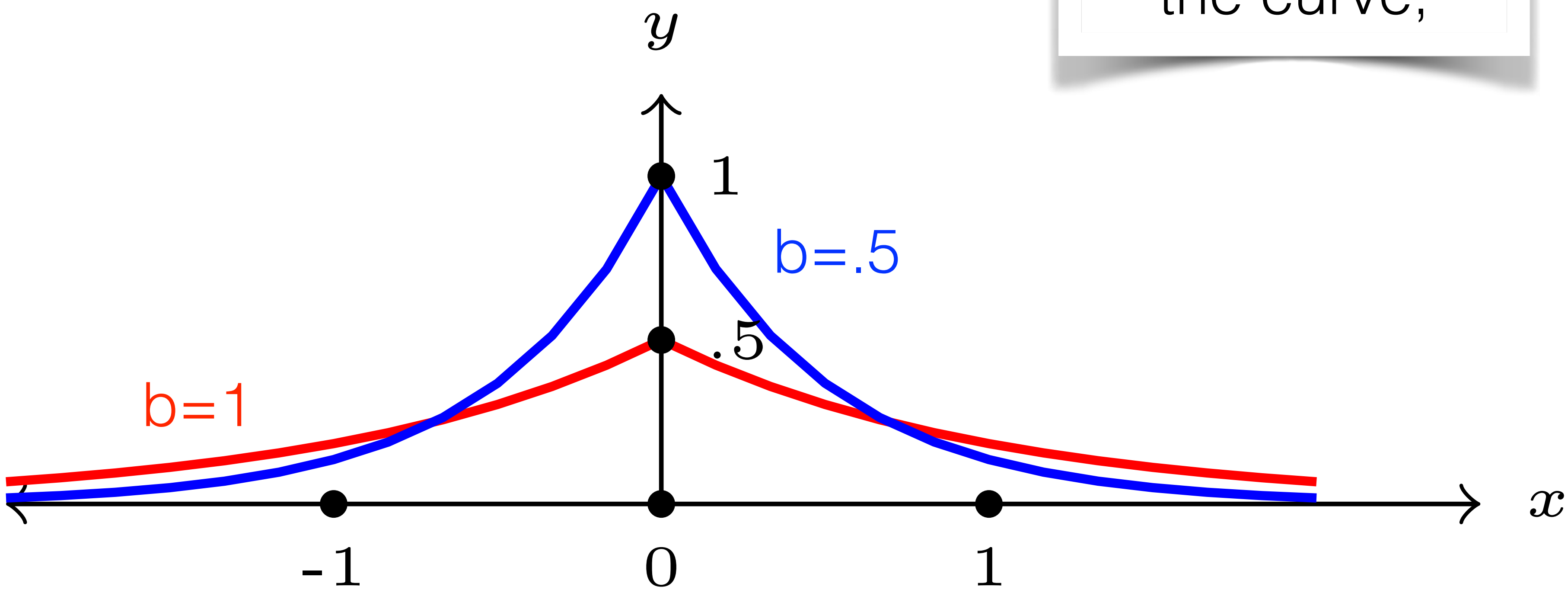
Notation $\texttt{r}_{\texttt{k}}, \texttt{r}_{\texttt{k}}'$ noise added at round k.

Let's focus on the iteration $\texttt{i}$ and let's fix the noises $\texttt{r}_{\texttt{j}}$ for all $\texttt{j} \neq \texttt{i}$

Laplace Distribution

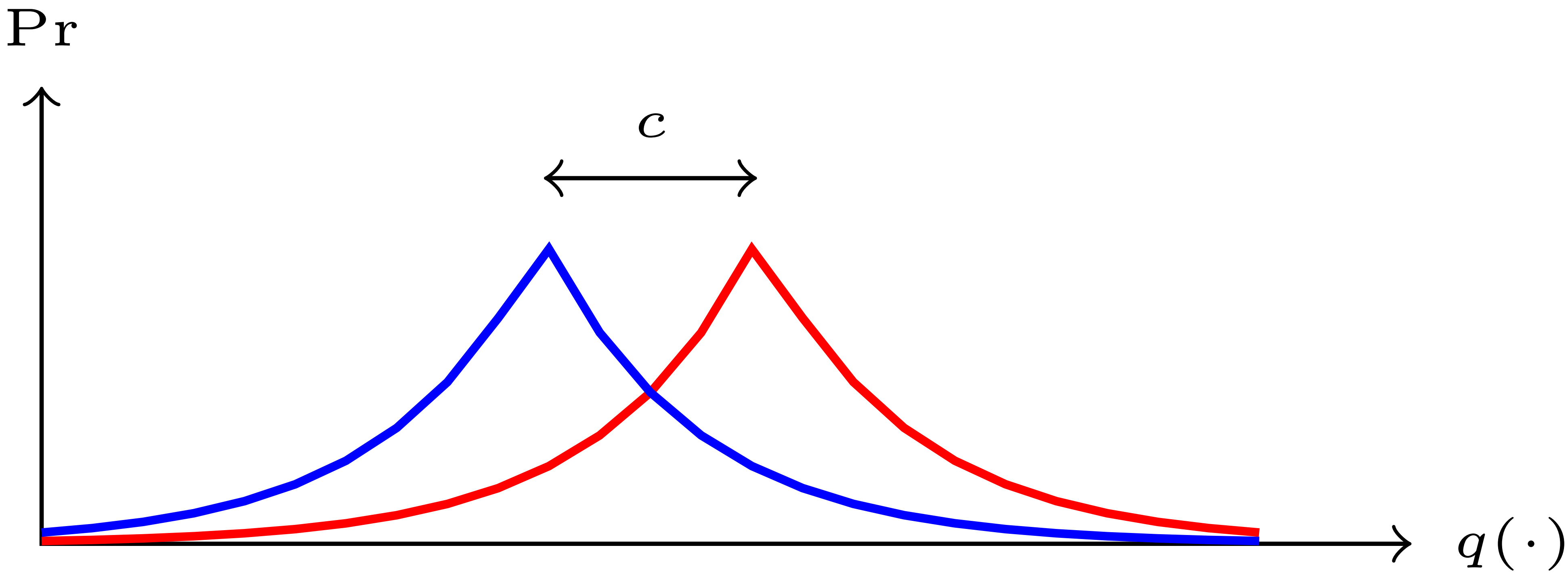
$$\mathsf{Lap}(b, \mu)(X) = \frac{1}{2b} \exp\left(- \frac{|\mu - X|}{b} \right)$$

b regulates the skewness of the curve,



Sliding property of the Laplace Distribution

$$\Pr_{x \sim Lap(\frac{1}{\epsilon},\mu)}[k \le x] \le e^{c\epsilon} \Pr_{x \sim Lap(\frac{1}{\epsilon},\mu)}[k + c \le x]$$



Report Noisy Max

Summarizing we have:

$$\Pr_{x \sim RNM(D)}[x = i \mid r_{-i}]$$

$$= \Pr_{r \sim Lap}[r \geq r^*] \leq e^{\epsilon} \Pr_{r \sim Lap}[r \geq 1 + r^*]$$

$$\leq e^{\epsilon} \Pr_{x \sim RNM(D')}[x = i \mid r_{-i}]$$

Report Noisy Max

In a similar way we can prove:

$$\Pr_{x \sim RNM(D')}[x = i \mid r_{-i}] \leq e^{\epsilon} \Pr_{x \sim RNM(D)}[x = i \mid r_{-i}]$$

Report Noisy Max

```
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
   i = 0;
   max = 0;
   while (i < N){
        cur = qi(b) + Lap(1/ε)
        if (cur > max)
              max =  cur ;
              output = i;
   return output;
```

Report Noisy Max

```
|-(ε,0)
[adj b1 b2,GS(qi)≤1,…]
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
  return output;
[output1=output2]
```

Point-wise reformulation of differential privacy

Given $\varepsilon, \delta \geq 0$, a mechanism M: db $\rightarrow O$ where O is discrete, is (ε,δ)-differentially private iff $\forall b_1 \sim_1 b_2$ and $\forall s \in O$:

$$\Pr[M(b_1) = s] \leq \exp(\varepsilon) \cdot \Pr[M(b_2) = s] + \delta_s$$

with $\sum \delta_s \leq \delta$.

Can we turn this definition into a rule?

apRHL: pointwise DP rule

$$\texttt{forall } r \in R$$

$$\vdash_{\varepsilon,\delta r} c_1 \sim c_2 : P \Longrightarrow x\langle 1\rangle = r \Rightarrow x\langle 2\rangle = r$$

$$\sum \delta r \leq \delta$$

$$\vdash_{\varepsilon,\delta} c_1 \sim c_2 : P \Longrightarrow x\langle 1\rangle = x\langle 2\rangle$$

Report Noisy Max

```
forall s, |-(ε,0)
[adj b1 b2,GS(qi)≤1,…]
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
     cur = qi(b) + Lap(1/ε)
     if (cur > max)
          max = cur ;
          output = i;
  return output;
[output1=s => output2=s]
```

By applying the pointwise rule we get a different post

Report Noisy Max

```
forall s, |-(ε,0)
[adj b₁ b₂,GS(qᵢ)≤1,…]
RNM (q₁,…,qN : (data → R) list,
      b : list data, ε: R) : nat
   i = 0;
   max = 0;
   while (i < N){
      cur = qᵢ(b) + Lap(1/ε)
      if (cur > max)
          max =  cur ;
          output = i;
   return output;
[output₁=s => output₂=s]
```

By applying the pointwise rule we get a different post

Notice that we focus on a single general s.

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
[adj b1 b2,GS(qi)≤1,…]
  while (i < N){
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

We can apply standard RHL

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, invariant]
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

Invariant

$$
\begin{aligned}
&\ldots\ (max_1 < cur_1 \Rightarrow output_1 = i_1) \\
&\wedge\ (max_2 < cur_2 \Rightarrow output_2 = i_2) \\
&\wedge\ i_1 = i_2\ \wedge\ output_1 = output_2
\end{aligned}
$$

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv]<fun k => if k=s then ε else 0>
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max =  cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv,(i1=s \/ i1<>s)] <fun k … >
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max = cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

We can now proceed by cases

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1=s] <fun k => if k=s then ε
else 0>
     cur = qi(b) + Lap(1/ε)
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
[output1=s => output2=s]
```

Case 1

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1=s] <ε>
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
            max =  cur ;
            output = i;
  return output;
[output1=s => output2=s]
```

We can simplify

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1=s] <ε>
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max = cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

What rule shall we apply now?

apRHL: More general Lap rule (still restricted)

$$\frac{}{\vdash_{k*\varepsilon,0} \texttt{x}_1 \texttt{:=\$ Lap(1/}\varepsilon\texttt{,y}_1\texttt{)} \sim \texttt{x}_2 \texttt{:=\$ Lap(1/}\varepsilon\texttt{,y}_2\texttt{)} : |\texttt{y}_1 - \texttt{y}_2| \le k \Rightarrow =}$$

Invariant

… $(max_1 < cur_1 \Rightarrow output_1 = i_1)$

$\wedge$ $(max_2 < cur_2 \Rightarrow output_2 = i_2)$

$\wedge$ $i_1 = i_2$ $\wedge$ $output_1 = output_2$

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
    cur = qi(b) + Lap(1/ε)
[adj b1 b2,GS(qi)≤1,…, inv, i1=s, cur1=cur2] <0>
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
[output1=s => output2=s]
```

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
   i = 0;
   max = 0;
   while (i < N){
      cur = qi(b) + Lap(1/ε)
[adj b1 b2,GS(qi)≤1,…, inv, i1=s, cur1=cur2] <0>
      if (cur > max)
            max = cur ;
            output = i;
   return output;
[output1=s => output2=s]
```

We can conclude this case by the asynchronous if rule

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1<>s] <fun k => if k=s then ε
else 0>
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
            max = cur ;
            output = i;
  return output;
[output1=s => output2=s]
```

Case 2

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
   i = 0;
   max = 0;
   while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1<>s] <0>
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
            max = cur ;
            output = i;
   return output;
[output1=s => output2=s]
```

We can simplify

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1<>s] <0>
     cur = qi(b) + Lap(1/ε)
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
[output1=s => output2=s]
```

What rule shall we apply now?

apRHL: More general Lap rule (still restricted)

$$\dfrac{}{\vdash_{k*\varepsilon,0} \begin{array}{l} \texttt{x}_1 := \$ \ \texttt{Lap}(1/\varepsilon, \texttt{y}_1) \\ \sim \\ \texttt{x}_2 := \$ \ \texttt{Lap}(1/\varepsilon, \texttt{y}_2) \\ : \ |\texttt{y}_1 - \texttt{y}_2| \le k \Rightarrow = \end{array}}$$

Can we use this rule here?

apRHL
Generalized Laplace

$$\vdash_{k2 * \varepsilon, 0} \quad \begin{array}{l} x_1 := \$ \ \mathrm{Lap}(\varepsilon, e_1) \\ \sim \\ x_2 := \$ \ \mathrm{Lap}(\varepsilon, e_2) \\ : \ |k_1 + e_1\langle 1\rangle - e_2\langle 2\rangle| \le k_2 \\ \qquad \Longrightarrow x1\langle 1\rangle + k_1 = x\langle 2\rangle \end{array}$$

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
[adj b1 b2,GS(qi)≤1,…, inv, i1<>s] <0>
      cur = qi(b) + Lap(1/ε)
      if (cur > max)
           max = cur ;
           output = i;
  return output;
[output1=s => output2=s]
```

We can apply this rule with $k_1=q_{i<2>}-q_{i<1>}$

Morally

```
|-(0,0) Pre: true
output = input + Lap(ε)
Post: [output1-output2=input1-input2]
```

Report Noisy Max

```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
     b : list data, ε: R) : nat
  i = 0;
  max = 0;
  while (i < N){
     cur = qi (b) + Lap(1/ε)
[adj b1 b2,GS(qi)≤1,…, inv, i1<>s, cur1+qi<2>-qi<1>=cur2] <0>
     if (cur > max)
          max =  cur ;
          output = i;
  return output;
[output1=s => output2=s]
```

We can apply this rule with $k_1 = q_{i<2>} - q_{i<1>}$

Report Noisy Max

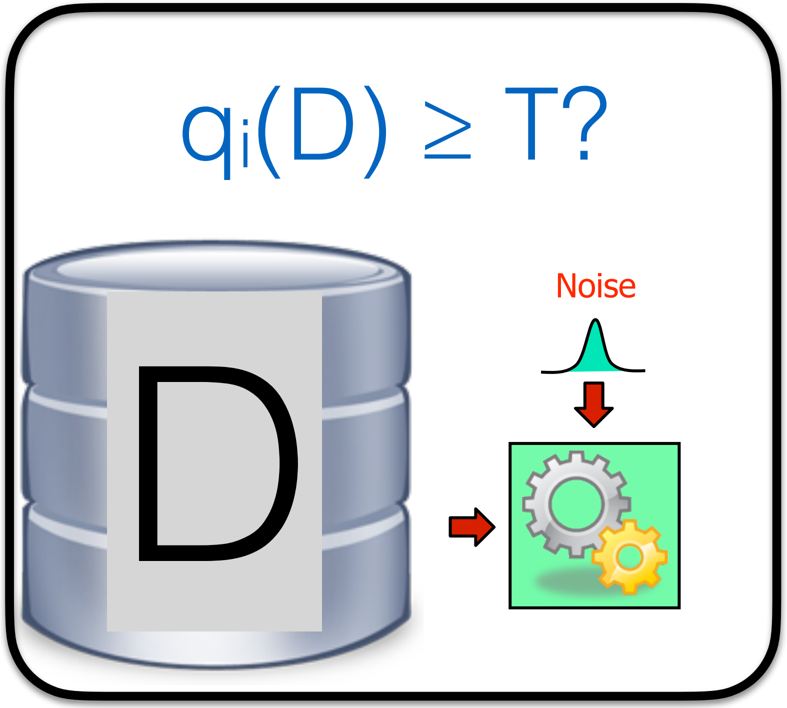
```
forall s, |-(ε,0)
RNM (q1,…,qN : (data → R) list,
      b : list data, ε: R) : nat
   i = 0;
   max = 0;
   while (i < N){
      cur = qi (b) + Lap(1/ε)
[adj b1 b2,GS(qi)≤1,…, inv, i1<>s, cur1+qi<2>-qi<1>=cur2] <0>
      if (cur > max)
            max =  cur ;
            output = i;
   return output;
[output1=s => output2=s]
```

We can conclude this case by the asynchronous if rule and conclude

One last example of differentially private programs

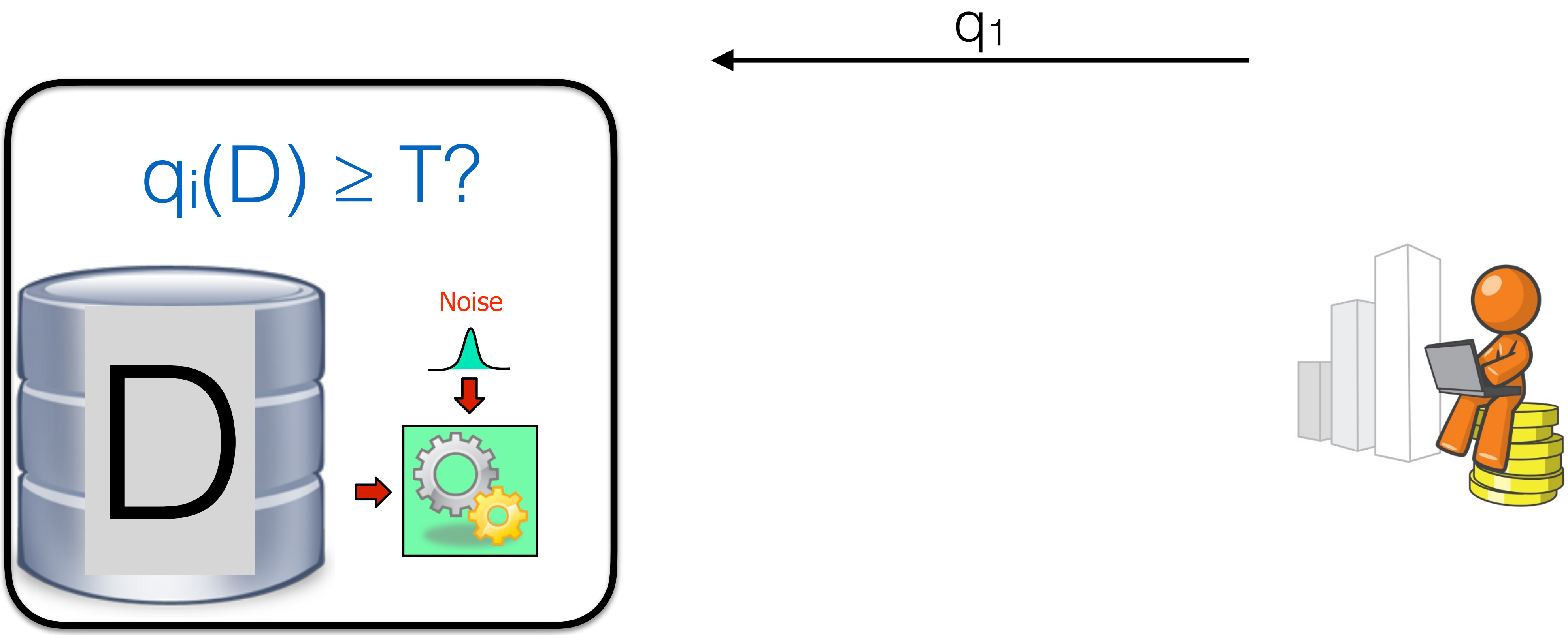
Sparse Vector

$$\mathrm{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



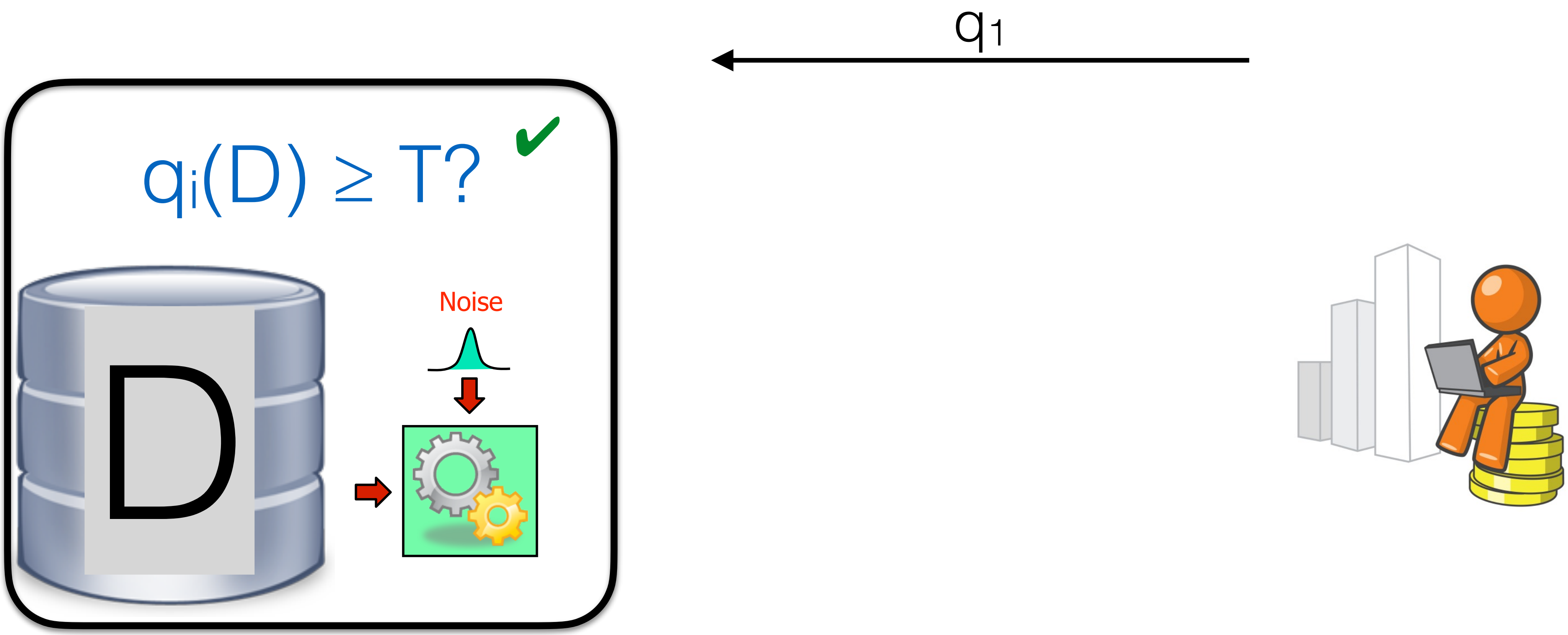
Sparse Vector

$$\mathrm{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



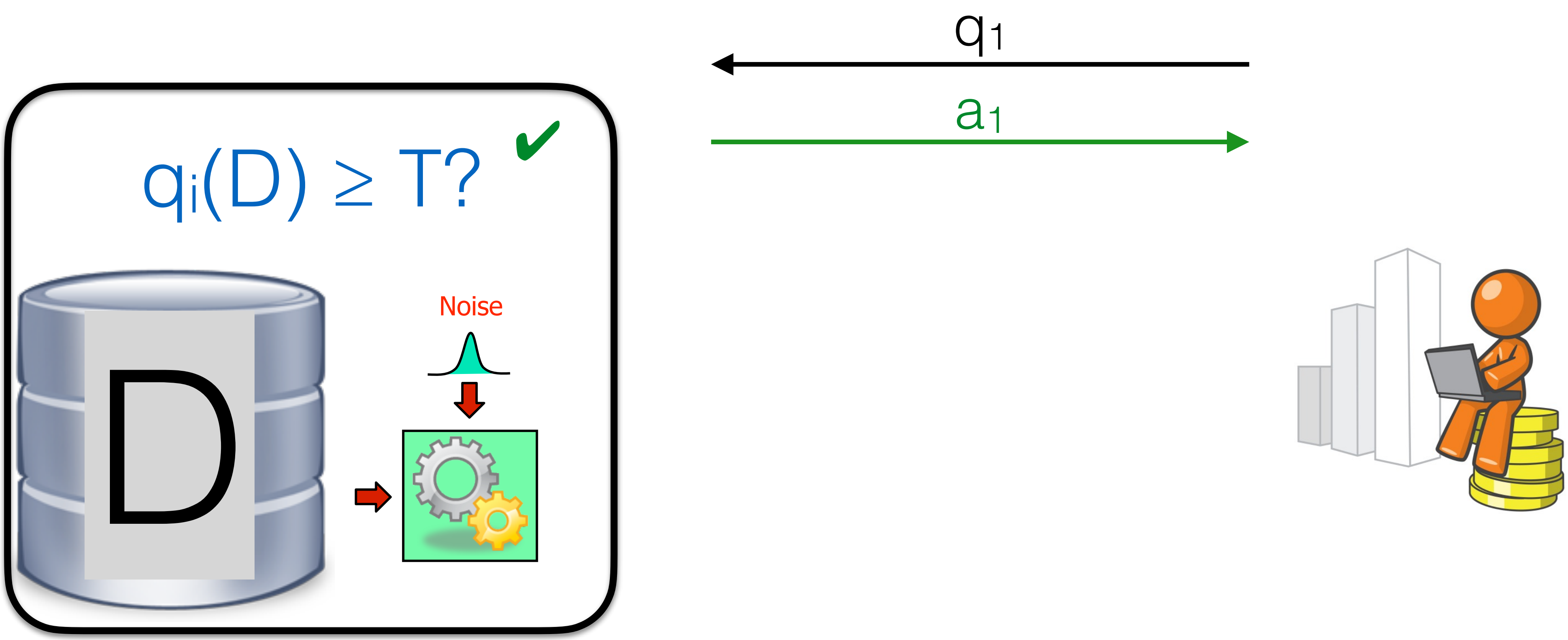
Sparse Vector

SparseVector(D,q_1,...,q_n,T,ε)



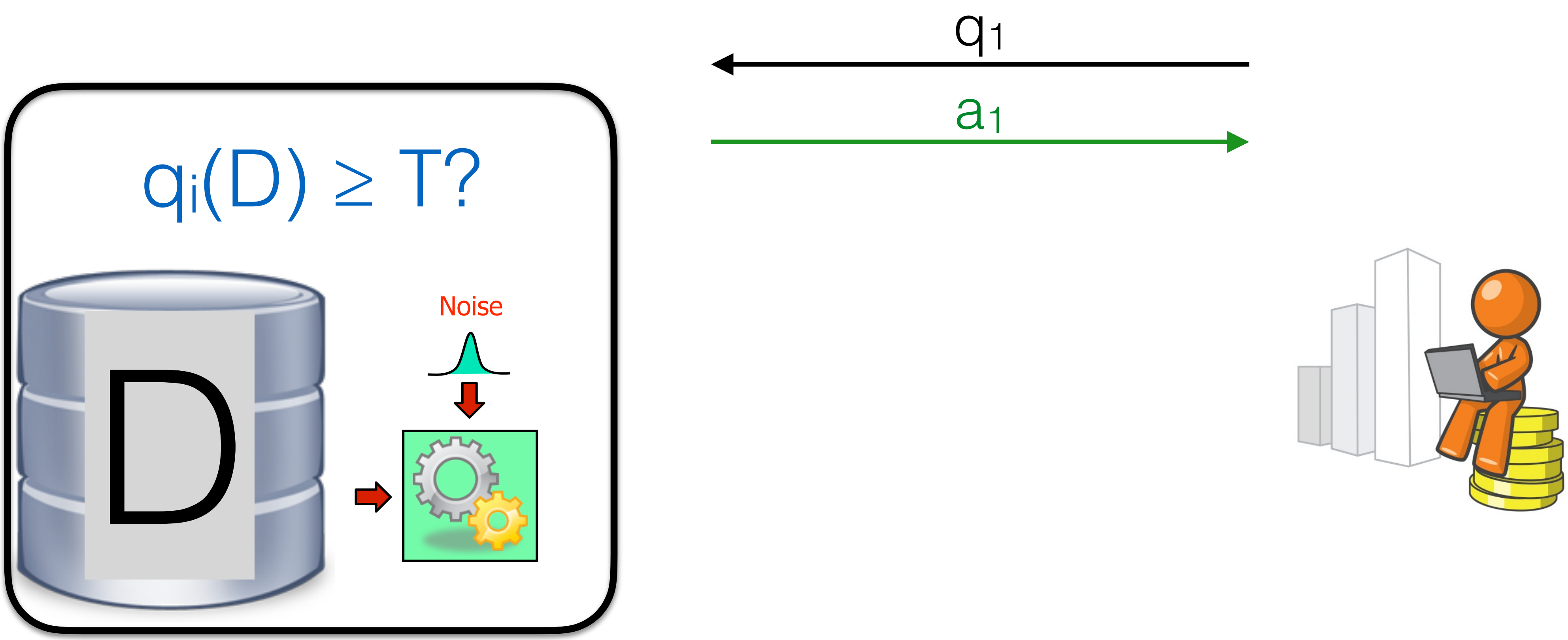
Sparse Vector

$$\mathrm{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



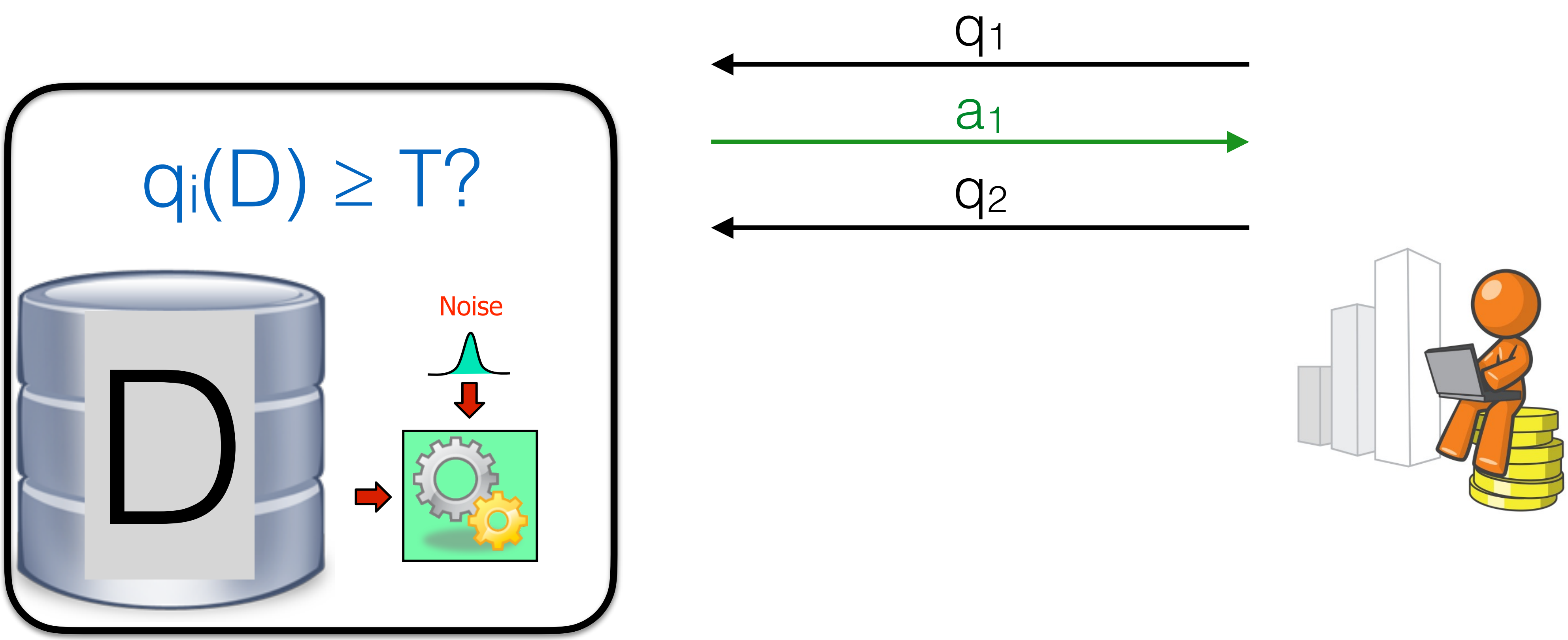
Sparse Vector

SparseVector($D, q_1, \ldots, q_n, T, \varepsilon$)



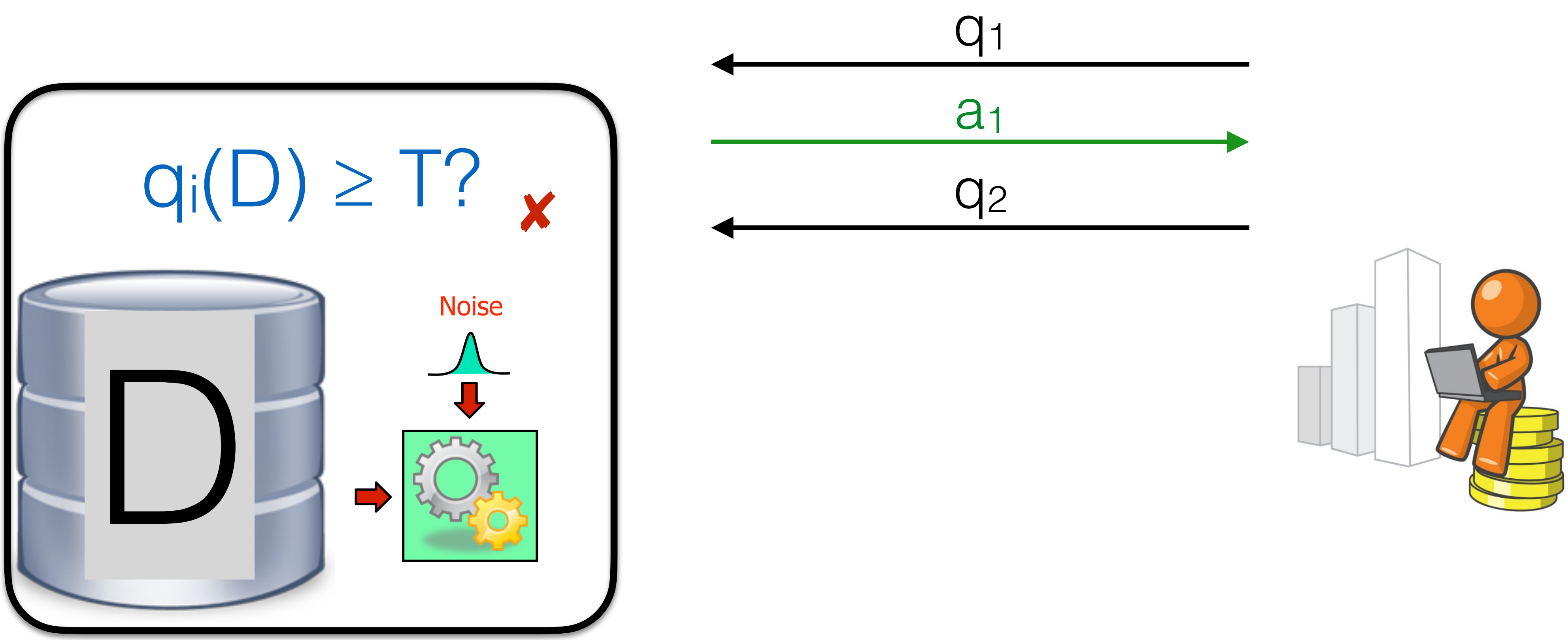
Sparse Vector

$$\mathrm{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



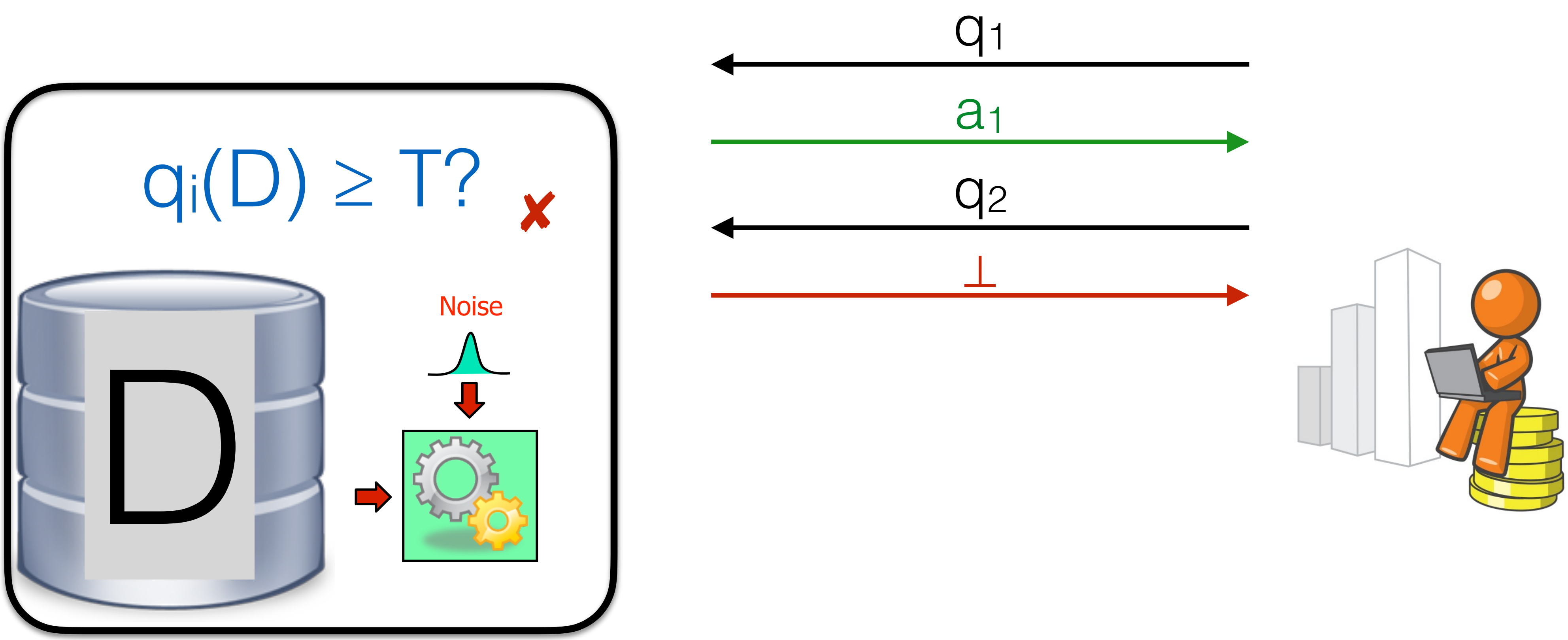
Sparse Vector

SparseVector(D,q1,...,qn,T,ε)



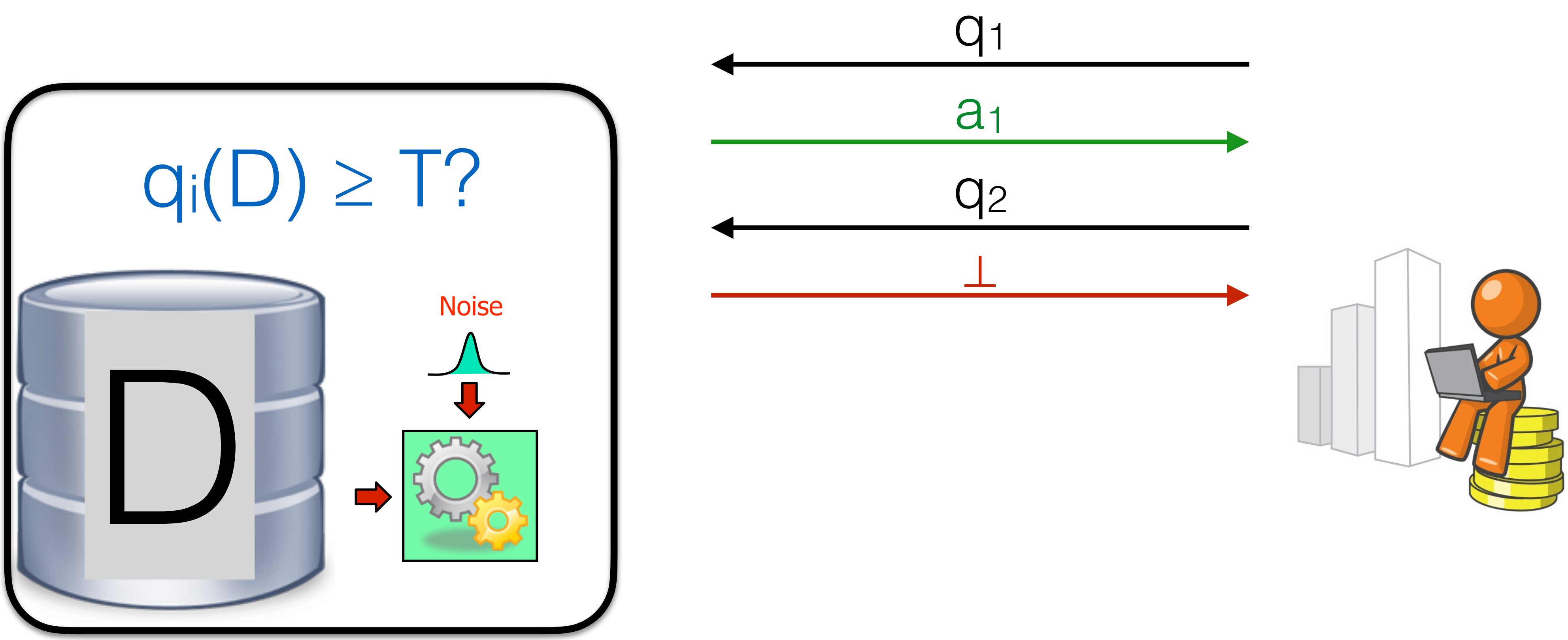
Sparse Vector

$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



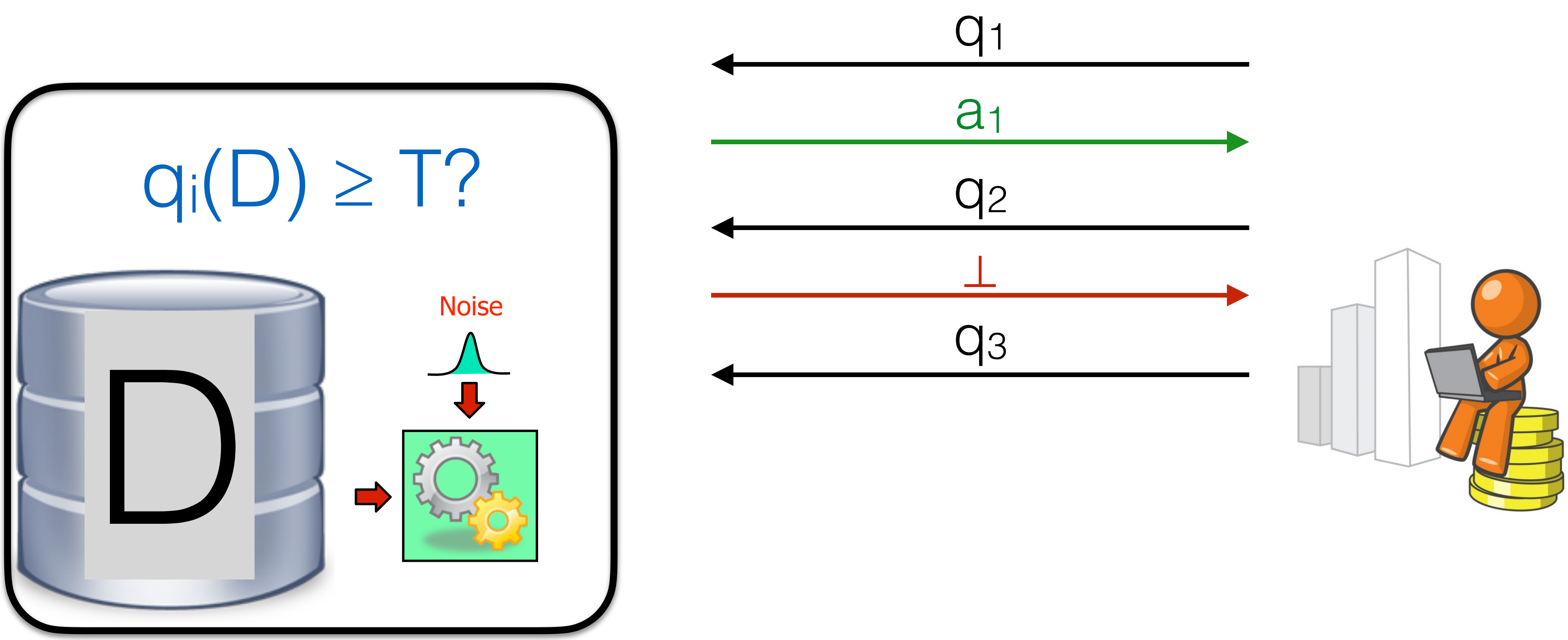
Sparse Vector

$$\mathrm{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



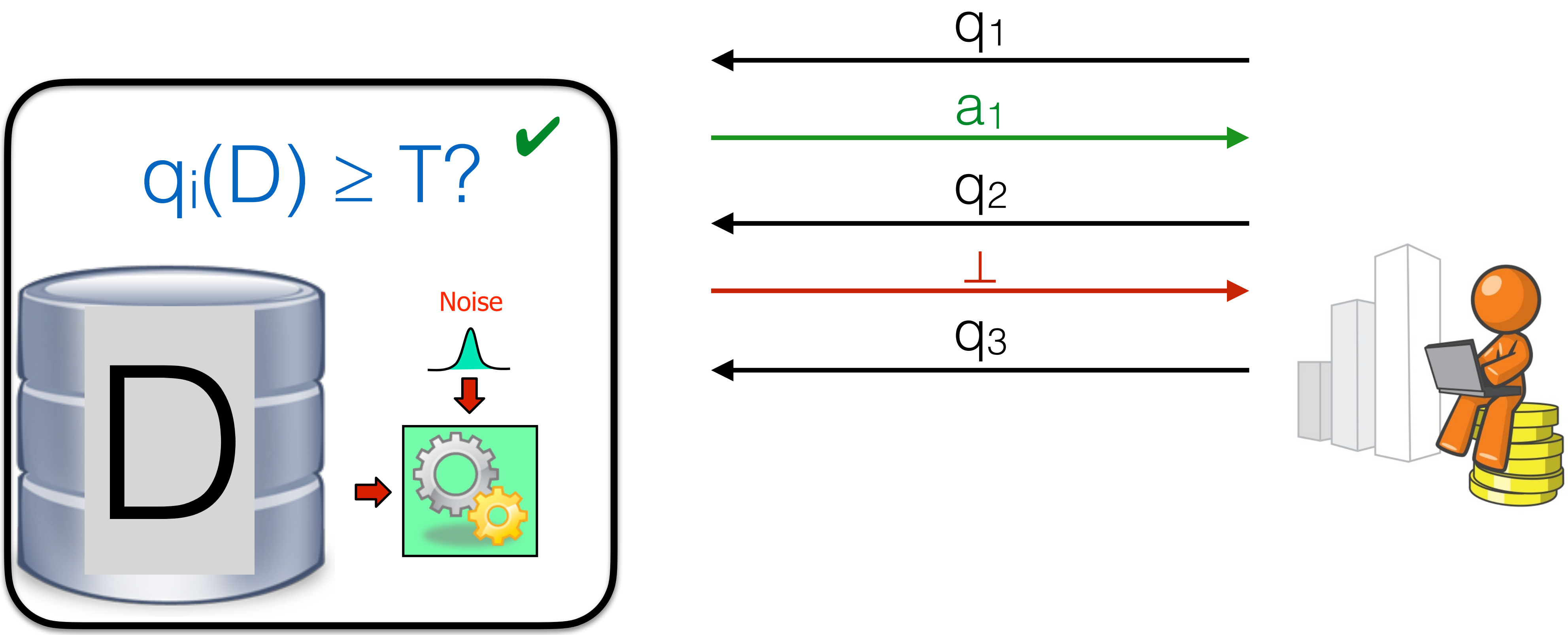
Sparse Vector

SparseVector(D,q_1,...,q_n,T,ε)



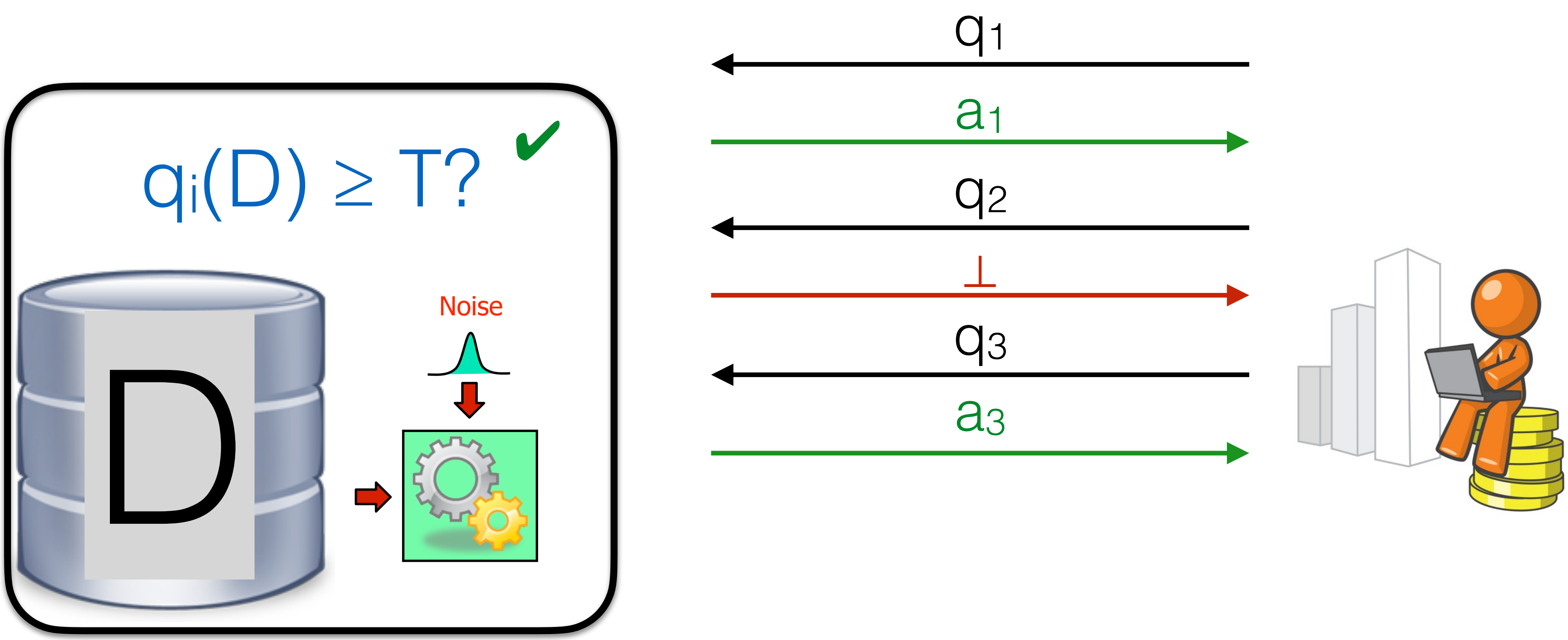
Sparse Vector

$$\mathrm{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



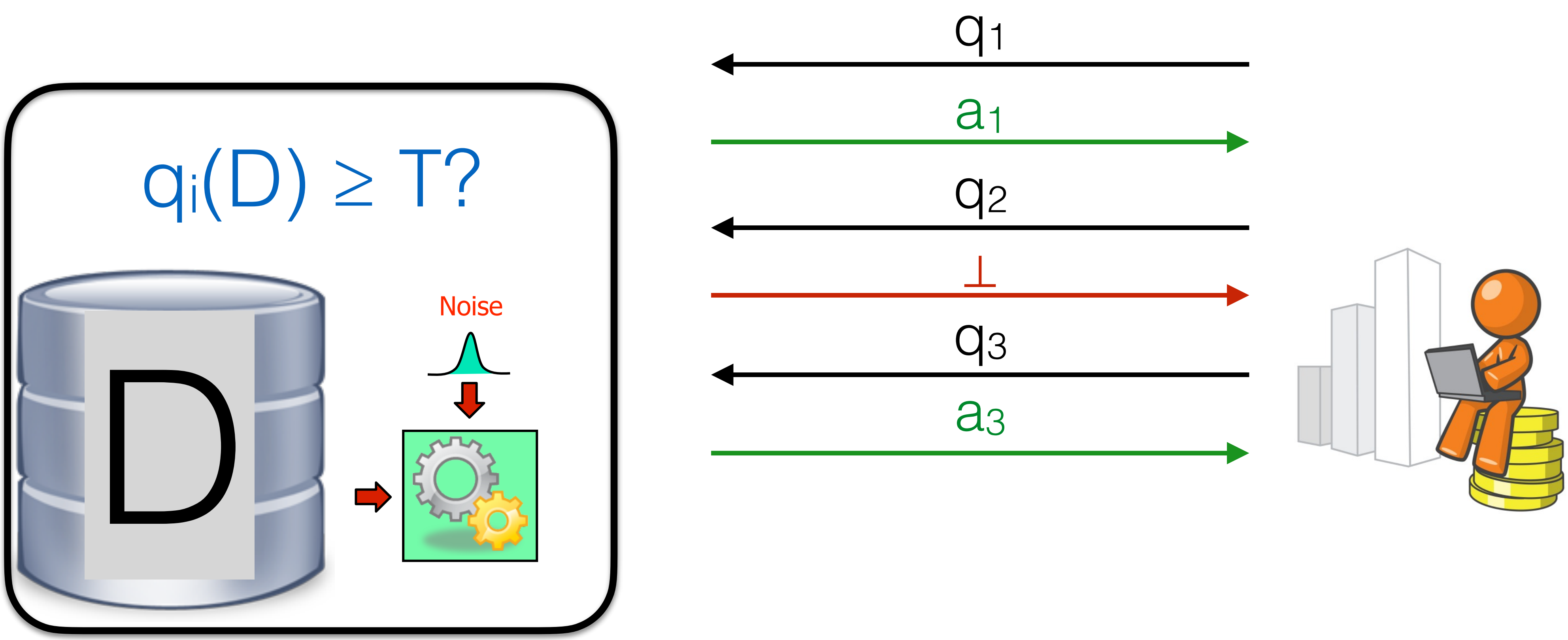
Sparse Vector

$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



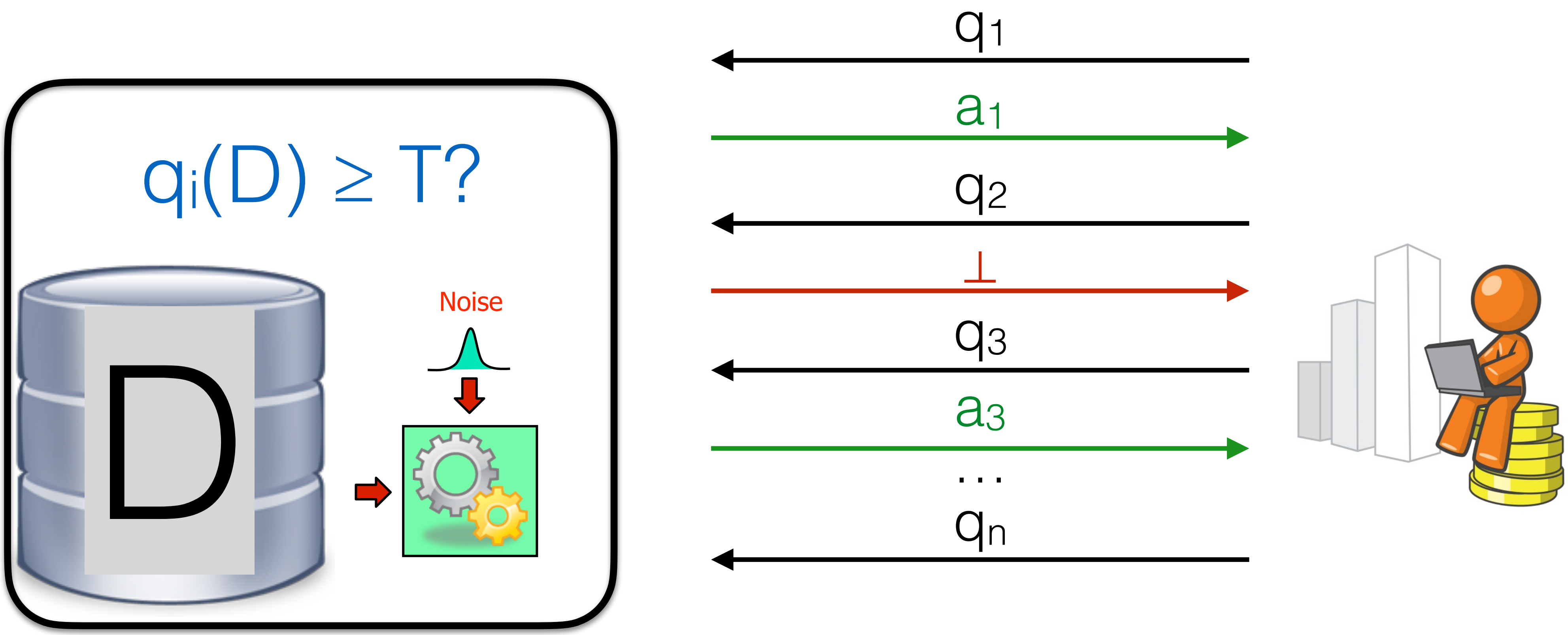
Sparse Vector

$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



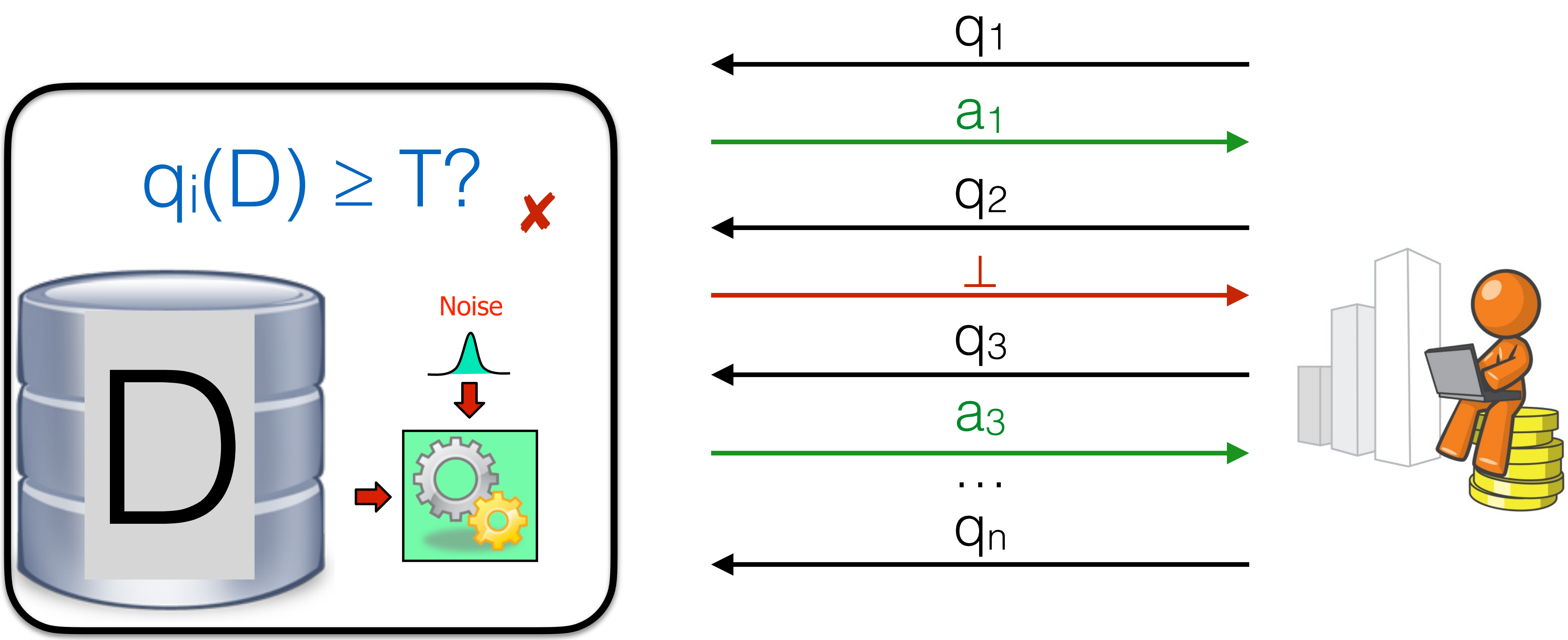
Sparse Vector

$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



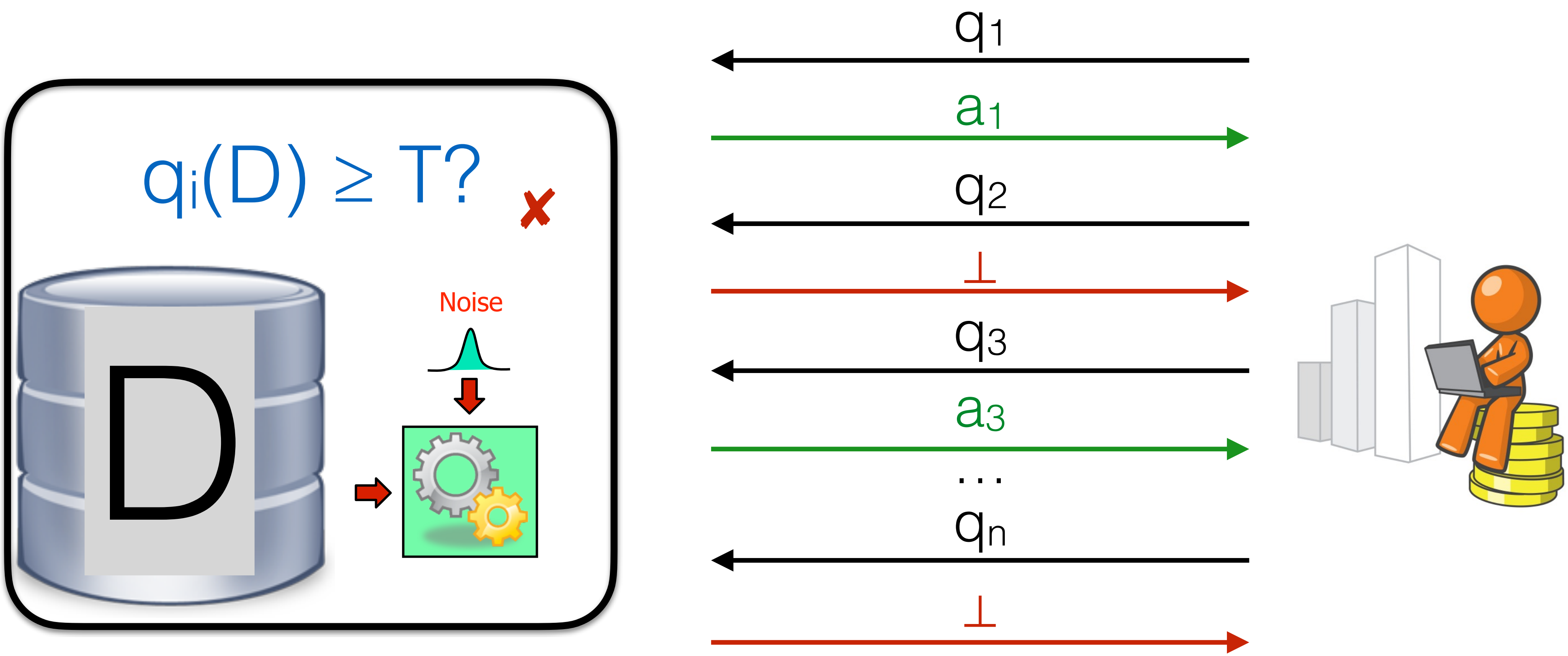
Sparse Vector

$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



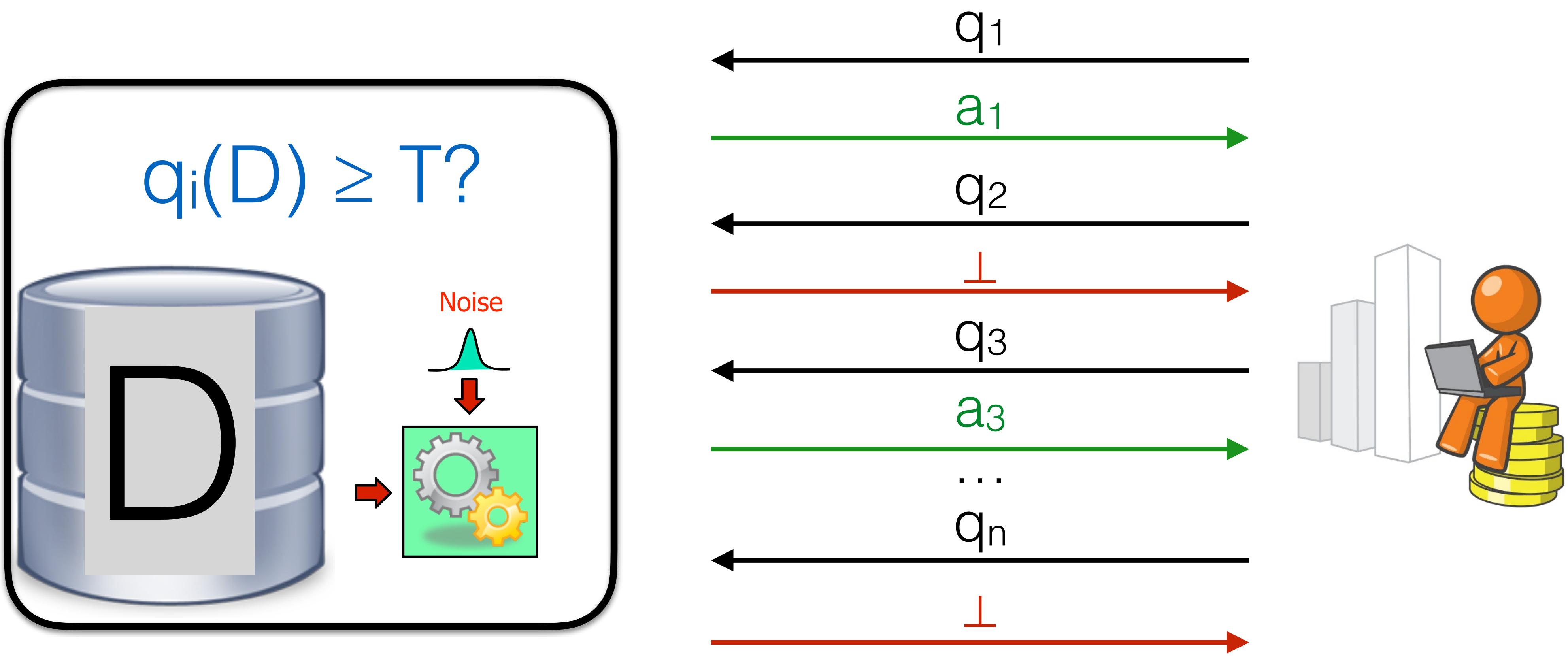
Sparse Vector

$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



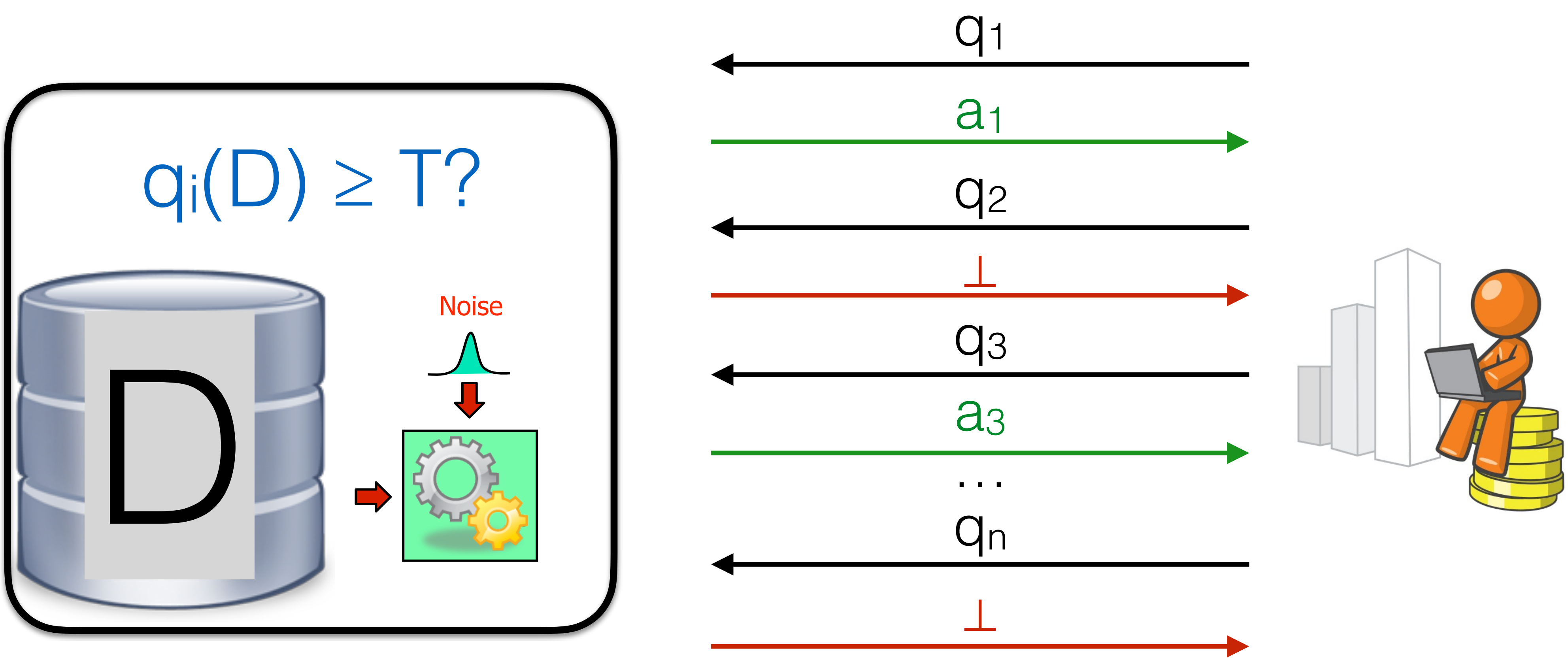
Sparse Vector

SparseVector(D, q_1, . . . , q_n, T, ε)



Sparse Vector

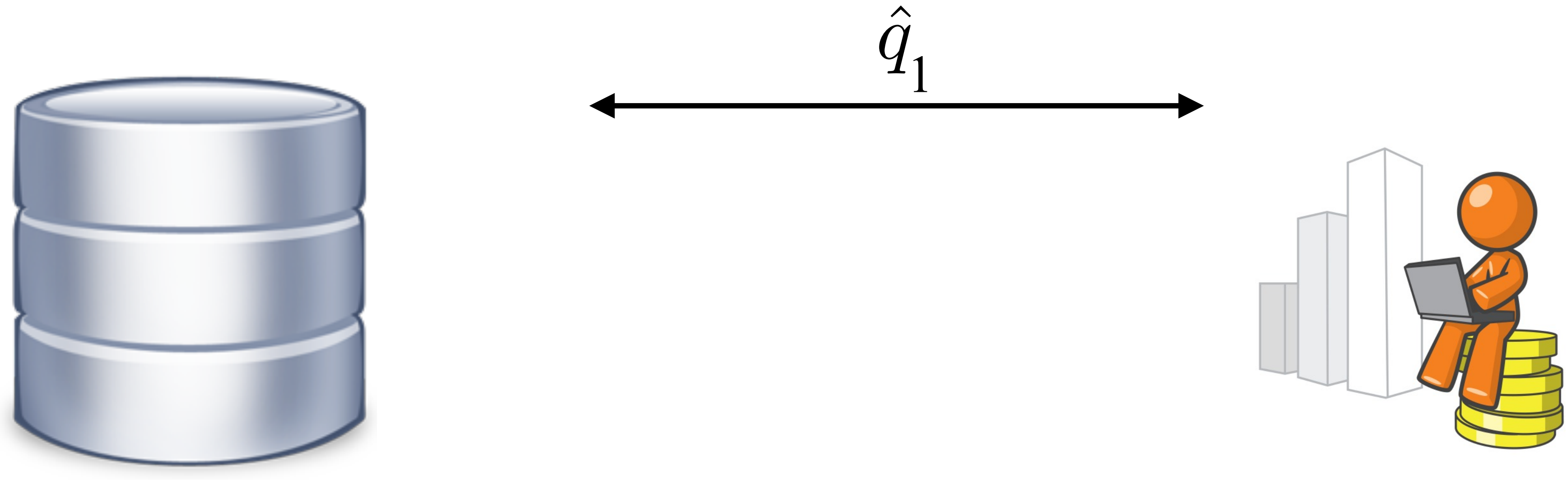
$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$



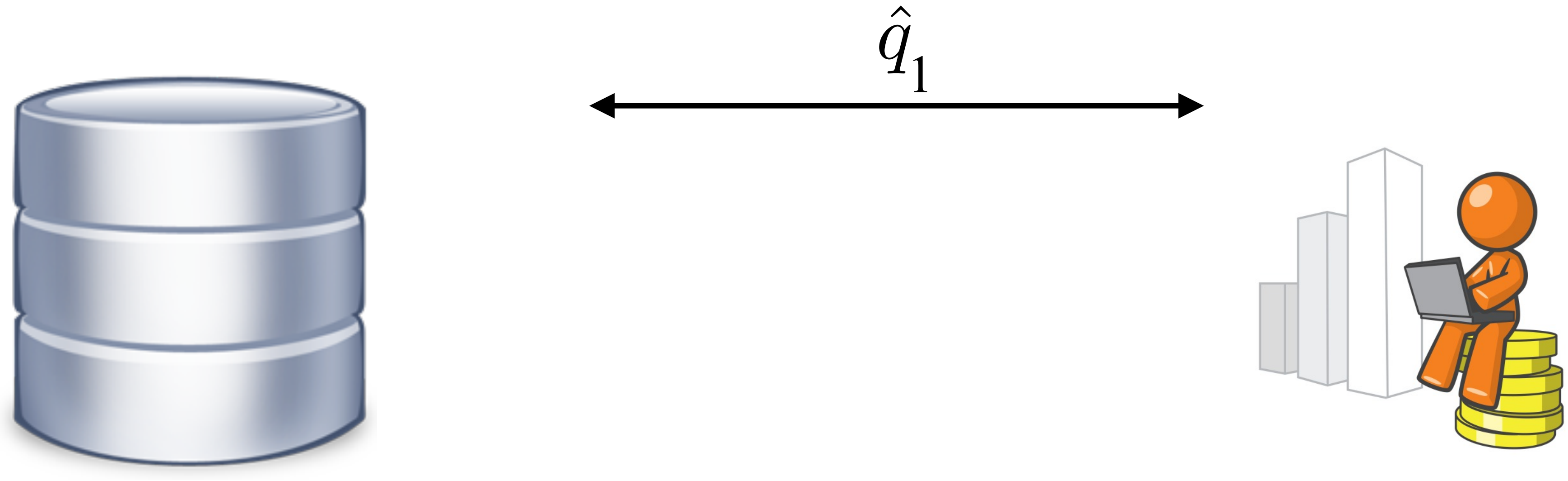
How can we achieve epsilon-DP by paying only for the queries above T?

A first step: above threshold

A first step: above threshold



A first step: above threshold



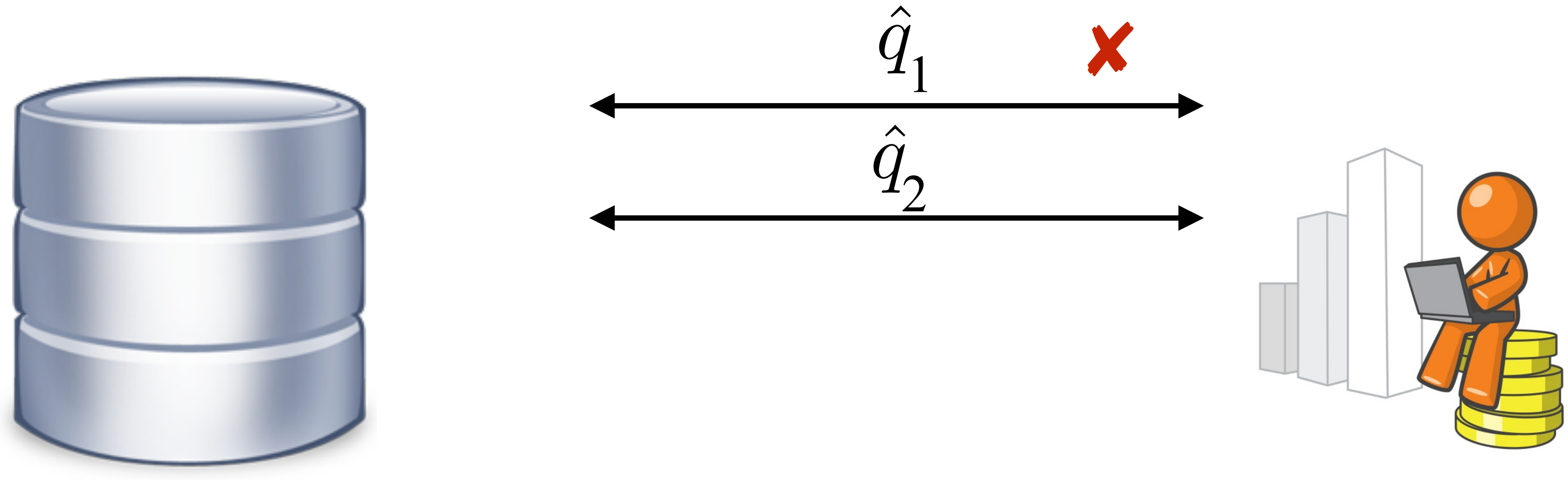
Above threshold $\hat{t}$?

A first step: above threshold

$\hat{q}_1$

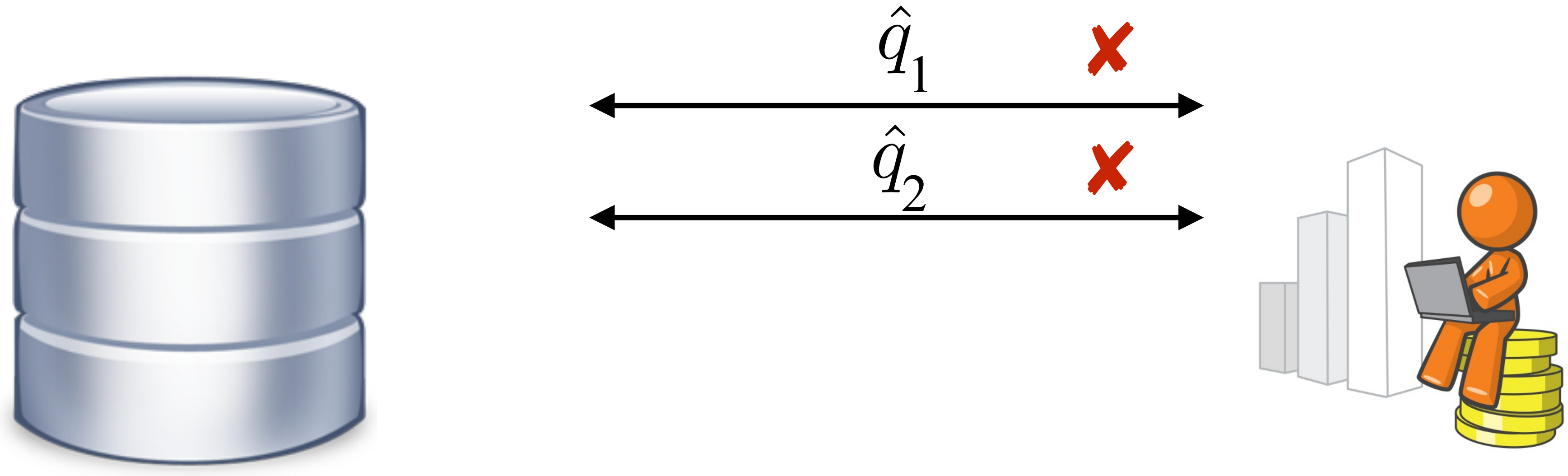
Above threshold $\hat{t}$?

A first step: above threshold



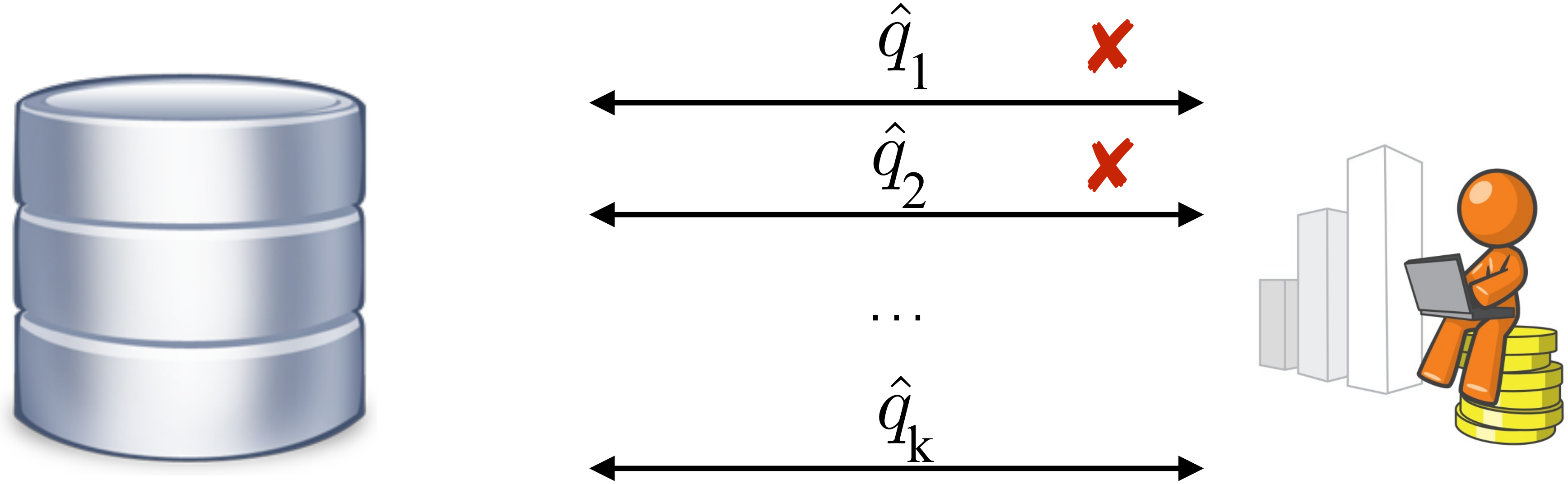
Above threshold $\hat{t}$?

A first step: above threshold



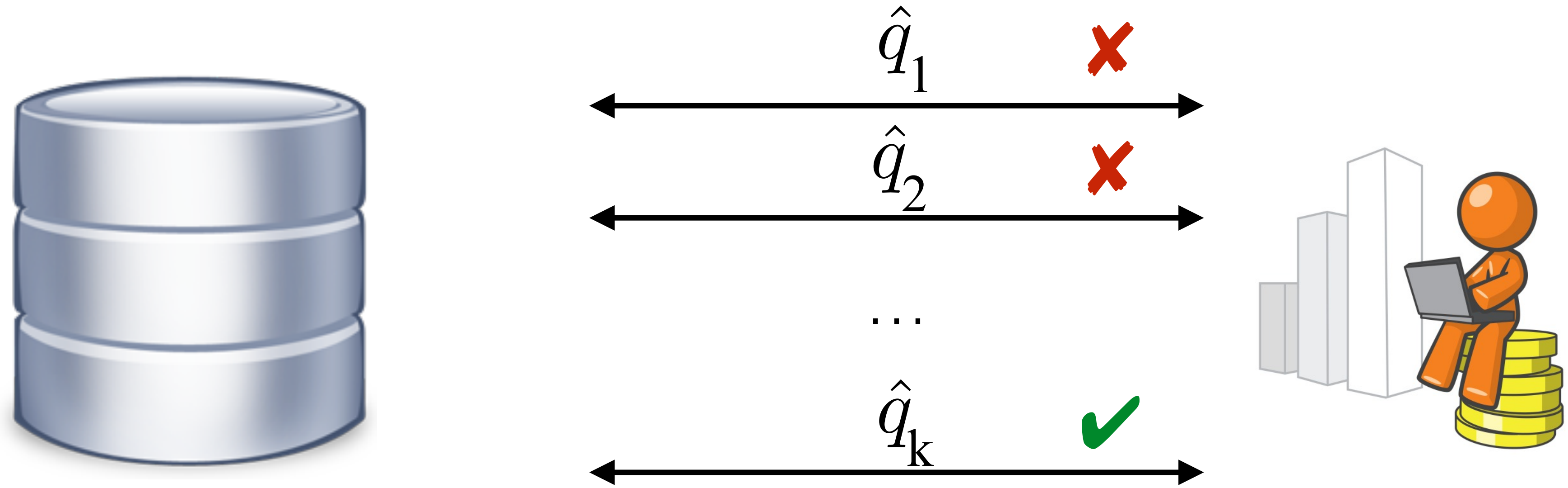
Above threshold $\hat{t}$?

A first step: above threshold



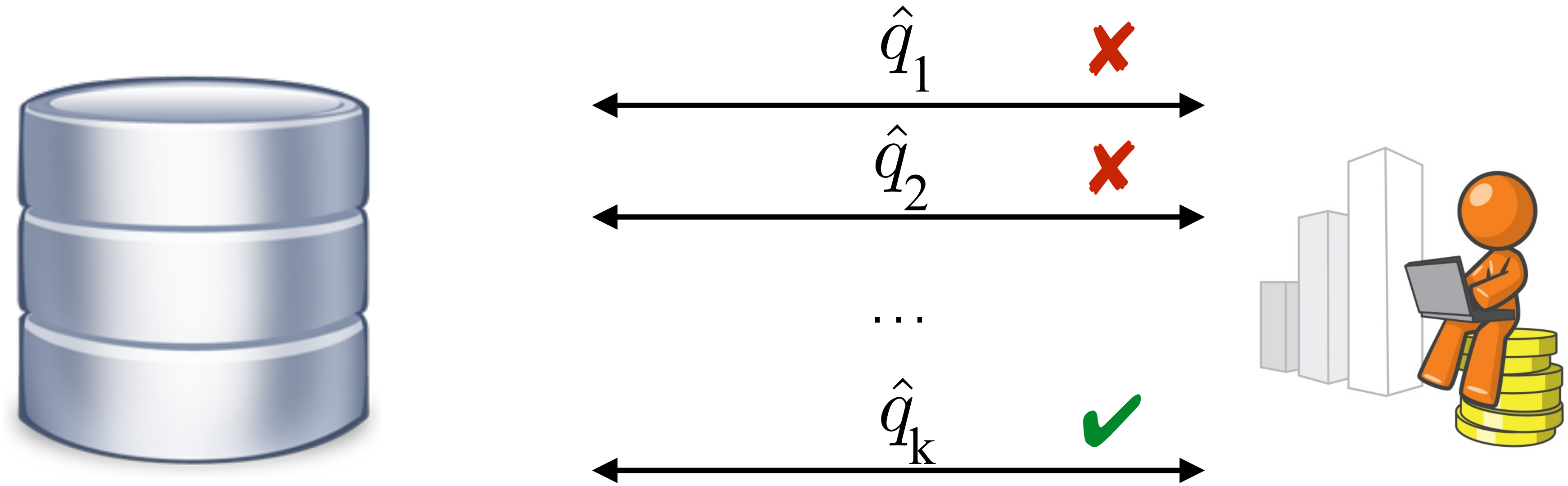
Above threshold $\hat{t}$?

A first step: above threshold



Above threshold $\hat{t}$?

A first step: above threshold



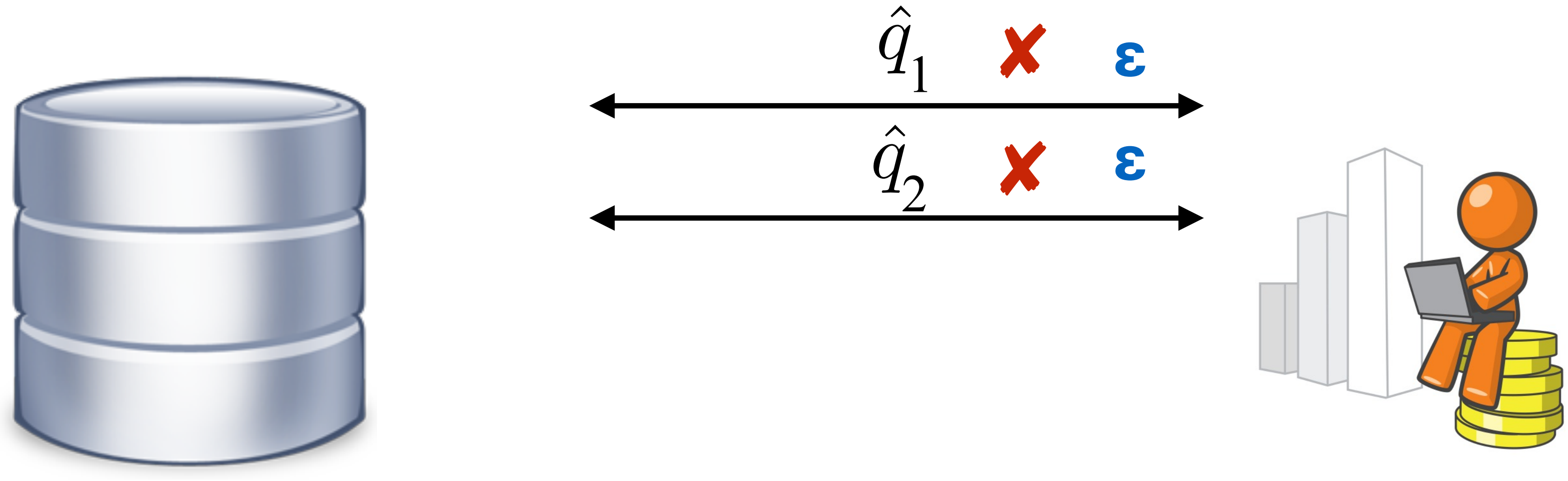
Return k

Reasoning by Composition

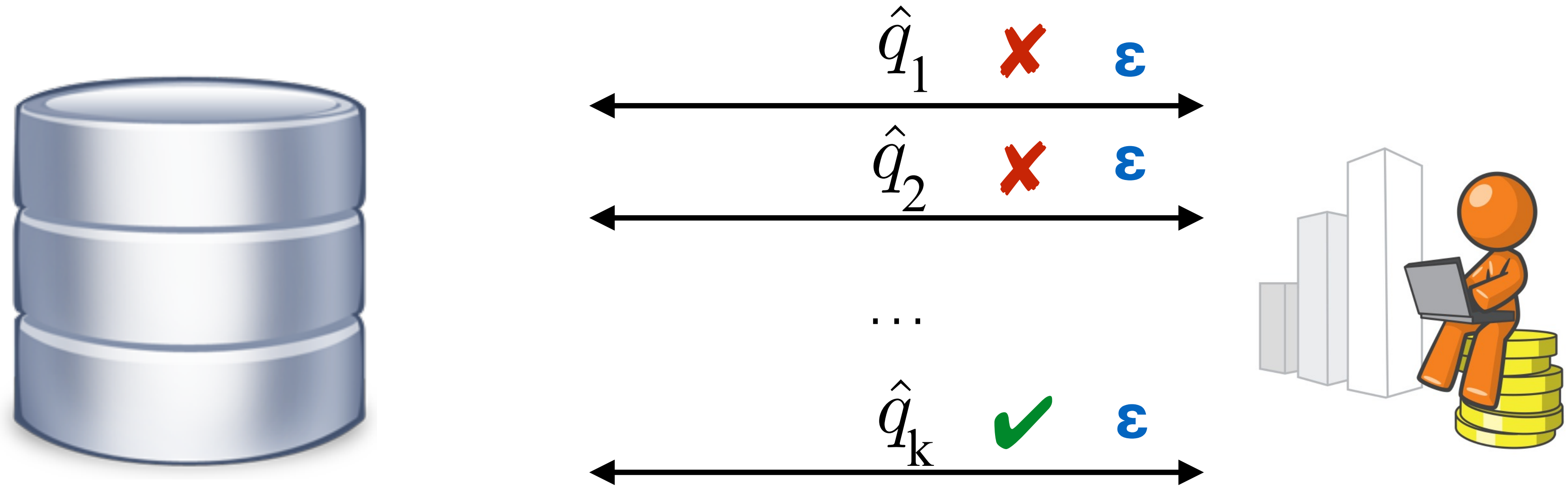
Reasoning by Composition

$\hat{q}_1$ ✘ ε

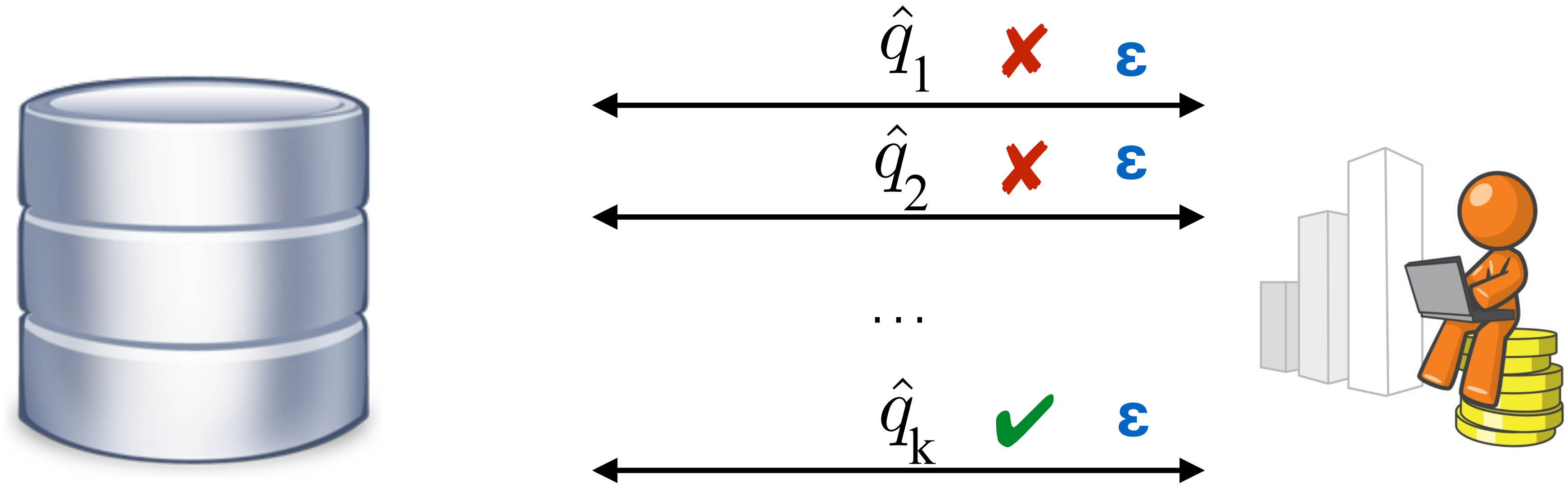
Reasoning by Composition



Reasoning by Composition

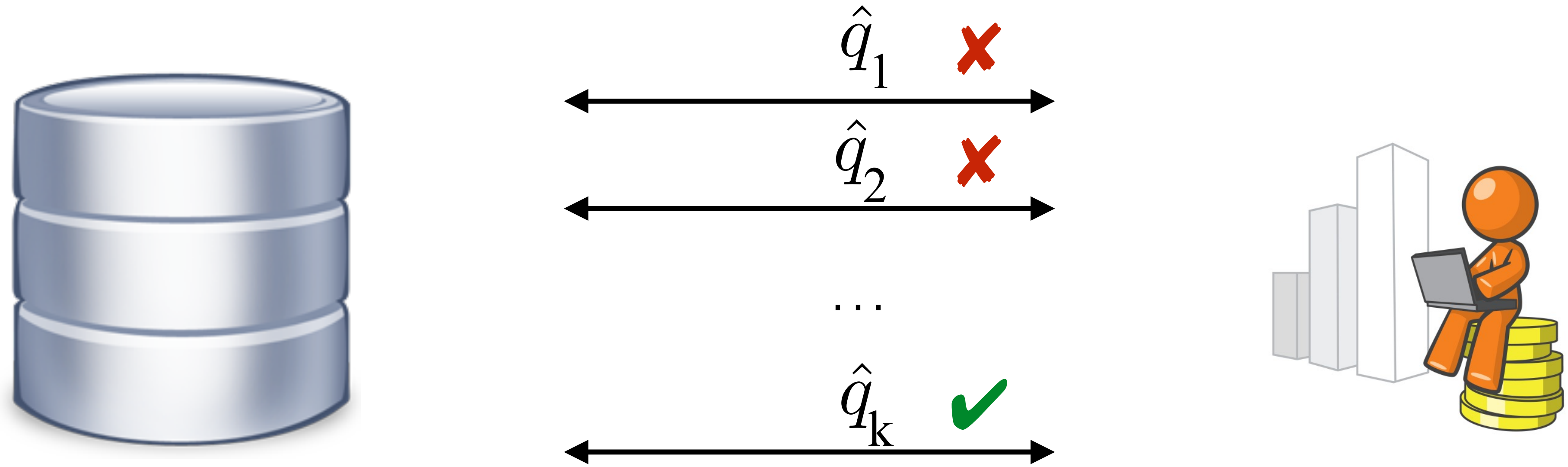


Reasoning by Composition

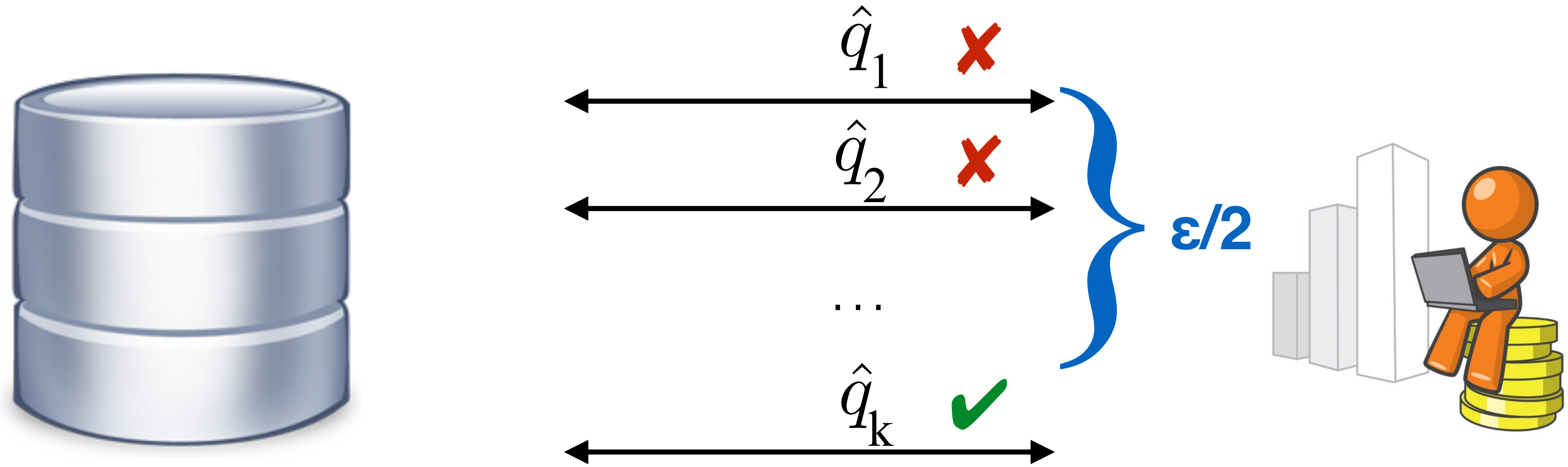


In the worst case,
the data analysis is (**nε**,0)-DP

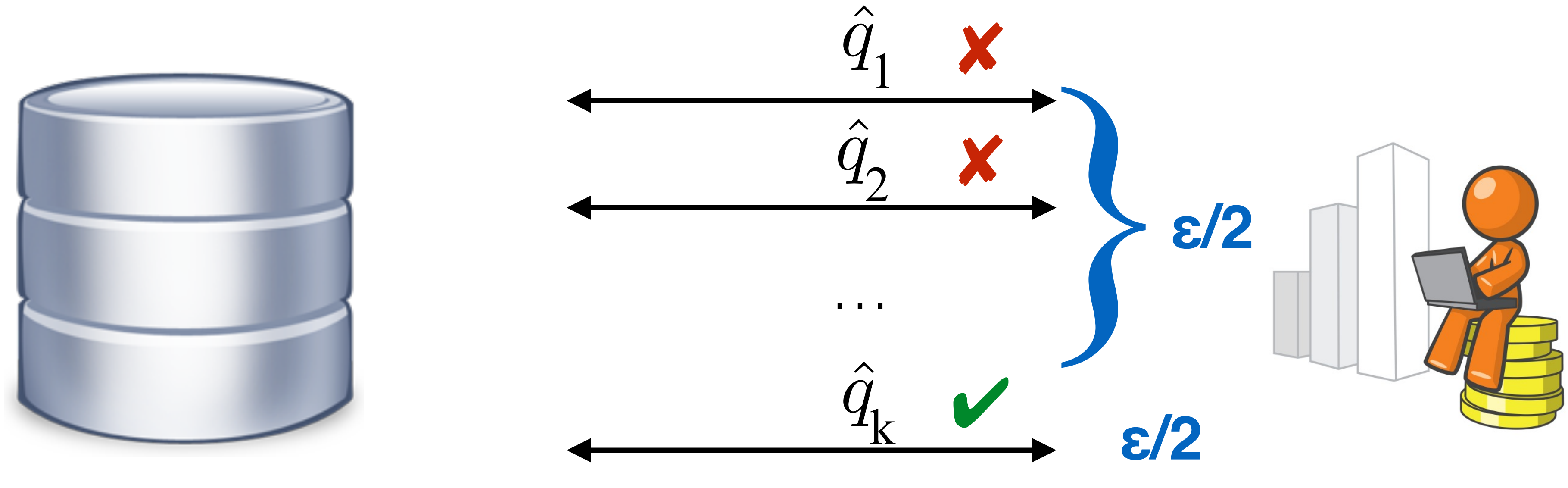
A more advanced analysis



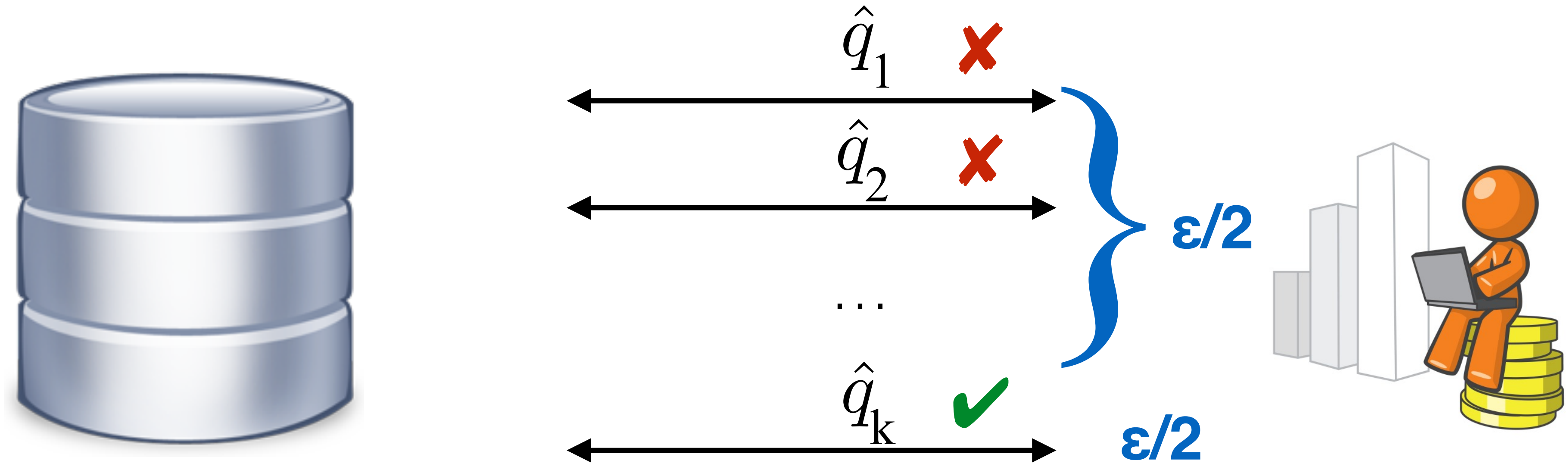
A more advanced analysis



A more advanced analysis

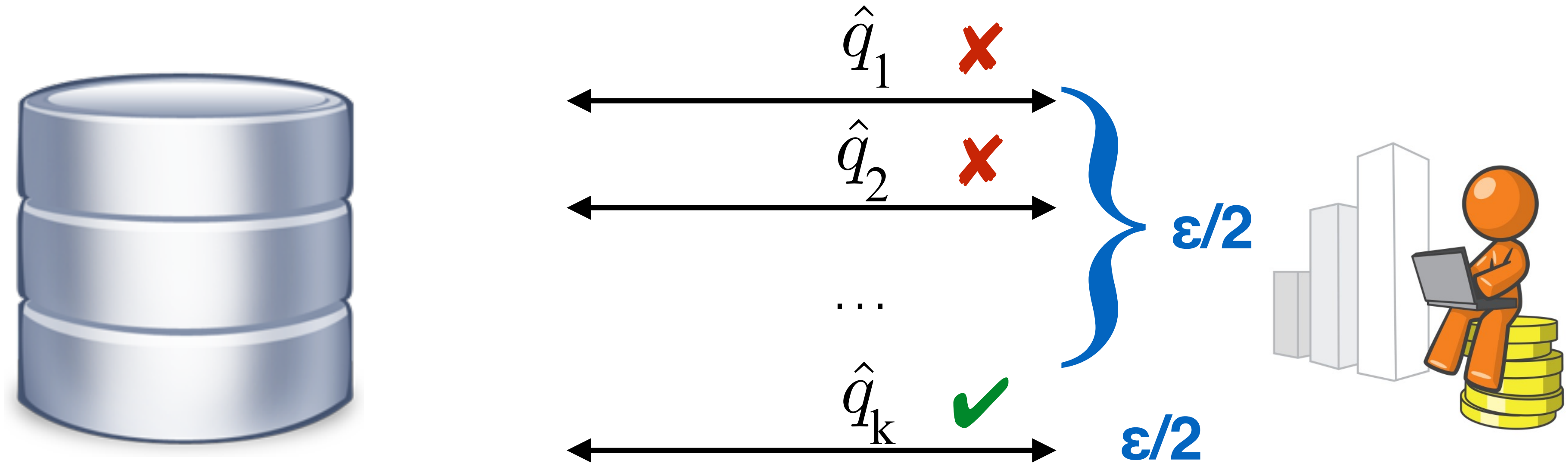


A more advanced analysis



We can show that above threshold is (**ε**,0)-DP

A more advanced analysis



We can show that above threshold is (**ε**,0)-DP

It doesn't depend on the number of queries.

Above Threshold

```
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
      if (cur ≥ nT /\ output = N )
         output = i;
      i++
  return output;
```

1-sensitive queries

Example 1: the sparse vector case

Algorithm 1 An instantiation of the SVT proposed in this paper.

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots, c$.

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1, \quad \text{count} = 0$
for each query $q_i \in Q$ **do**
    $\nu_i = \mathsf{Lap}\,(2c\Delta/\epsilon_2)$
    if $q_i(D) + \nu_i \geq T_i + \rho$ **then**
        Output $a_i = \top$
        count = count + 1, **Abort** if count $\geq c$.
    else
        Output $a_i = \perp$

Algorithm 2 SVT in Dwork and Roth 2014 [8].

Input: D, Q, Δ, T, c.

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(c\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1, \quad \text{count} = 0$
for each query $q_i \in Q$ **do**
    $\nu_i = \mathsf{Lap}\,(2c\Delta/\epsilon_1)$
    if $q_i(D) + \nu_i \geq T + \rho$ **then**
        Output $a_i = \top, \rho = \mathsf{Lap}\,(c\Delta/\epsilon_2)$
        count = count + 1, **Abort** if count $\geq c$.
    else
        Output $a_i = \perp$

Algorithm 3 SVT in Roth's 2011 Lecture Notes [15].

Input: D, Q, Δ, T, c.

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1),$
$\epsilon_2 = \epsilon - \epsilon_1, \quad \text{count} = 0$
for each query $q_i \in Q$ **do**
    $\nu_i = \mathsf{Lap}\,(c\Delta/\epsilon_2)$
    if $q_i(D) + \nu_i \geq T + \rho$ **then**
        Output $a_i = q_i(D) + \nu_i$
        count = count + 1, **Abort** if count $\geq c$.
    else
        Output $a_i = \perp$

Algorithm 4 SVT in Lee and Clifton 2014 [13].

Input: D, Q, Δ, T, c.

$\epsilon_1 = \epsilon/4, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1, \quad \text{count} = 0$
for each query $q_i \in Q$ **do**
    $\nu_i = \mathsf{Lap}\,(\Delta/\epsilon_2)$
    if $q_i(D) + \nu_i \geq T + \rho$ **then**
        Output $a_i = \top$
        count = count + 1, **Abort** if count $\geq c$.
    else
        Output $a_i = \perp$

Algorithm 5 SVT in Stoddard et al. 2014 [18].

Input: D, Q, Δ, T.

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$
for each query $q_i \in Q$ **do**
    $\nu_i = 0$
    if $q_i(D) + \nu_i \geq T + \rho$ **then**
        Output $a_i = \top$
    else
        Output $a_i = \perp$

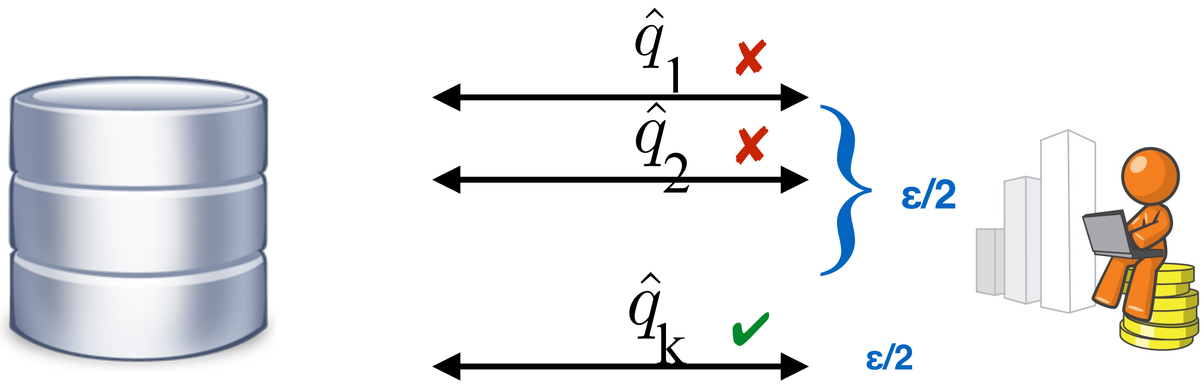
Algorithm 6 SVT in Chen et al. 2015 [1].

Input: $D, Q, \Delta, \mathbf{T} = T_1, T_2, \cdots$.

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$
for each query $q_i \in Q$ **do**
    $\nu_i = \mathsf{Lap}\,(\Delta/\epsilon_2)$
    if $q_i(D) + \nu_i \geq T_i + \rho$ **then**
        Output $a_i = \top$
    else
        Output $a_i = \perp$

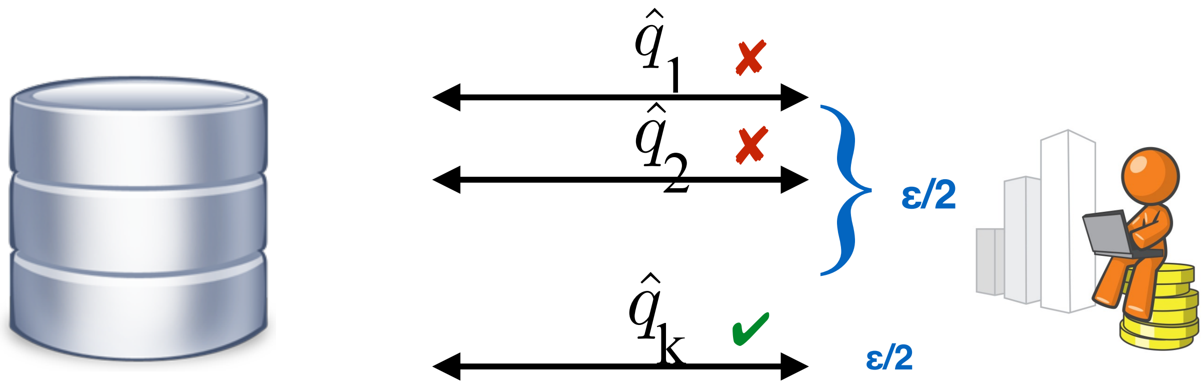
Min Lyu, Dong Su, Ninghui Li:
Understanding the Sparse Vector Technique for Differential Privacy. PVLDB (2017)

Above Threshold



Notation $\texttt{r}_\texttt{j}$
noise added at
round `j`.

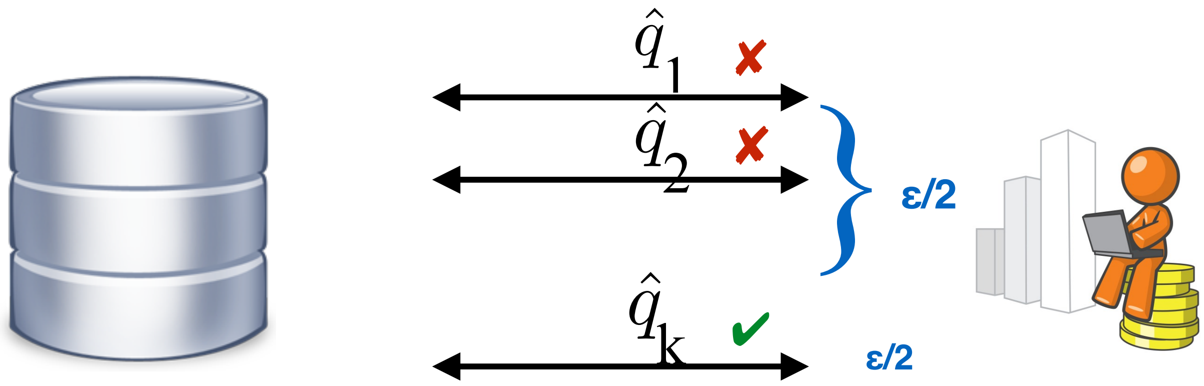
Above Threshold



Notation r_j noise added at round j.

Let's focus on k and let's fix the noises r_j for all $j \leq k$

Above Threshold



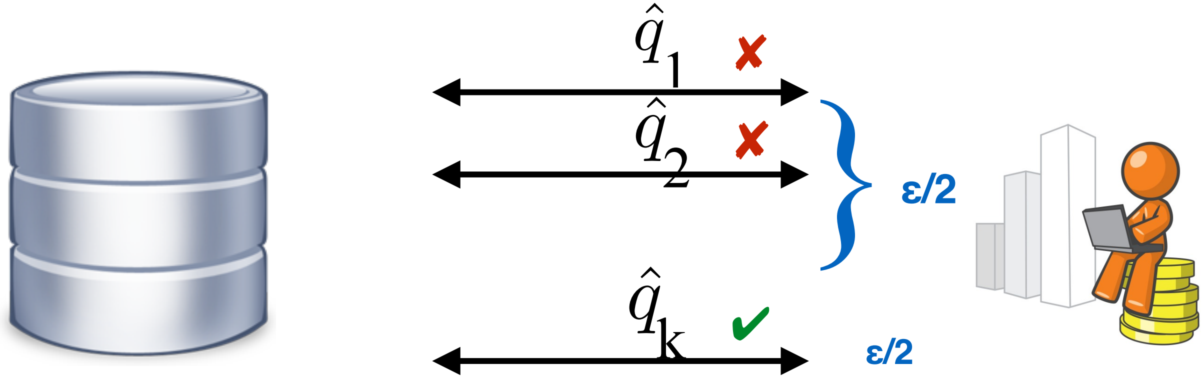
Notation $\texttt{r}_\texttt{j}$ noise added at round $\texttt{j}$.

Let's focus on $\texttt{k}$ and let's fix the noises $\texttt{r}_\texttt{j}$ for all $\texttt{j} \leq \texttt{k}$

We want to show:

$$\Pr_{x \sim AT(D)}[x = k \,|\, r_{-k}] \leq e^{\epsilon} \Pr_{x \sim AT(D')}[x = k \,|\, r_{-k}]$$

Above Threshold



Notation r_j noise added at round j.

Let's focus on k and let's fix the noises r_j for all $j \le k$

We want to show:

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] \le e^{\epsilon} \Pr_{x \sim AT(D')}[x = k \mid r_{-k}]$$

By fixing the noises r_j for all $j \le k$ we can compute the following quantity

$$\boxed{g(D) = \max_{i<k} \; q_i(D) + r_i}$$

Above Threshold

$$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

Above Threshold

$$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$$

$$\Pr_{x \sim AT(D)}[x = k \,|\, r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \,|\, r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k)]$$

Above Threshold

$$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k)]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

Above Threshold

$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k)]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

$$\leq exp(\frac{\epsilon}{2} * 1 + \frac{\epsilon}{4} * 2) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \mid r_{-k}]$$

Above Threshold

$$\boxed{g(D) = \max_{i<k} \; q_i(D) + r_i}$$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] =$$

Above Threshold

$$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

Above Threshold

$$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k) \mid r_{-k}]$$

Above Threshold

$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$

$$\Pr_{x \sim AT(D)}[x = k \,|\, r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \,|\, r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k) \,|\, r_{-k}]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$nT' = nT + g(D) - g(D')$

Above Threshold

$$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$$

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k) \mid r_{-k}]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

$$\leq exp(\frac{\epsilon}{2} * 1 + \frac{\epsilon}{4} * 2) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \mid r_{-k}]$$

Above Threshold

$\boxed{g(D) = \max_{i<k} q_i(D) + r_i}$

$$\Pr_{x \sim AT(D)}[x = k \,|\, r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \,|\, r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k) \,|\, r_{-k}]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

$$\leq exp(\frac{\epsilon}{2} * 1 + \frac{\epsilon}{4} * 2) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \,|\, r_{-k}]$$

$$= exp(\epsilon) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \,|\, r_{-k}] = exp(\epsilon) \Pr_{x \sim AT(D')}[x = k \,|\, r_{-k}]$$

Above Threshold

```
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
      if (cur ≥ T /\ output = N )
         output = i;
      i++
  return output;
```

Above Threshold

$\vdash_{(\varepsilon,0)}$
[adj b_1 b_2,GS(q_i)≤1,…]

```
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
      if (cur ≥ T /\ output = N )
          output = i;
      i++
  return output;
```

[$output_1$=$output_2$]

Above Threshold

```
forall k, |-(ε,0)
[adj b1 b2,GS(qi)≤1,…]
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

By applying the pointwise rule we get a different post

Above Threshold

```
forall k, |-(ε,0)
[adj b1 b2,GS(qi)≤1,...]
AboveT (q1,...,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
      if (cur ≥ T /\ output = N )
          output = i;
      i++
  return output;
[output1=k => output2=k]
```

By applying the pointwise rule we get a different post

Notice that we focus on a single general k.

Above Threshold

```
forall k, |-(ε,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
[adj b1 b2,GS(qi)≤1,…]
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
      if (cur ≥ T /\ output = N )
         output = i;
      i++
  return output;
[output1=k => output2=k]
```

We can apply standard RHL

Above Threshold

```
forall k, |-(ε,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
[adj b1 b2,GS(qi)≤1,…]
  nT = T + Lap(2/ε)
  while (i < N){
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Which rule shall we apply?

apRHL
Generalized Laplace

$$\overline{}$$

$$\texttt{x}_1\texttt{:=\$ Lap(}\varepsilon\texttt{,e}_1\texttt{)}$$

$$\vdash_{\texttt{k2*}\varepsilon\texttt{,0}} \quad \sim$$

$$\texttt{x}_2\texttt{:=\$ Lap(}\varepsilon\texttt{,e}_2\texttt{)}$$

$$\texttt{: |k}_1\texttt{+e}_1\texttt{<1>-e}_2\texttt{<2>|}\leq\texttt{k}_2$$

$$\texttt{==> x1<1>+k}_1\texttt{=x<2>}$$

Above Threshold

$\boxed{\texttt{g(D)=max}_{\texttt{i<k}}\ \texttt{q}_{\texttt{i}}\texttt{(D)+r}_{\texttt{i}}}$

$$\Pr_{x \sim AT(D)}[x = k \,|\, r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \,|\, r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k) \,|\, r_{-k}]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

$$\leq exp(\frac{\epsilon}{2} * 1 + \frac{\epsilon}{4} * 2) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \,|\, r_{-k}]$$

$$= exp(\epsilon) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \,|\, r_{-k}] = exp(\epsilon) \Pr_{x \sim AT(D')}[x = k \,|\, r_{-k}]$$

Above Threshold

```
forall k, |-(ε/2,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1]
 while (i < N){
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Using $k_1=1$

Above Threshold

```
forall k, |-(ε/2,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant]
  <[fun x => if x=k then ε/2 else 0]>
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Using $k_1=1$

Above Threshold

```
forall k, |-(ε/2,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,(i1=k \/ i1<>k)]
  <[fun x => if x=k then ε/2 else 0]>
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

We can now proceed by cases

Above Threshold

```
forall k, |-(ε/2,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1=k]
  &lt;[fun x => if x=k then ε/2 else 0]&gt;
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Case 1

Above Threshold

```
forall k, |-(ε/2,0)
AboveT (q1,…,qk : list data → R,
            db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1=k]
  <[ε/2]>
      cur = qi(db) + Lap(4/ε)
      if (cur ≥ T /\ output = N )
         output = i;
      i++
  return output;
[output1=k => output2=k]
```

Case 1

Above Threshold

```
forall k, |-(ε/2,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1=k]
  <[ε/2]>
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Which rule shall we apply?

apRHL

Generalized Laplace

$$\frac{}{\vdash_{k2*\varepsilon,0} \begin{array}{l} \texttt{x}_1\texttt{:=\$ Lap}(\varepsilon,\texttt{e}_1) \\ \sim \\ \texttt{x}_2\texttt{:=\$ Lap}(\varepsilon,\texttt{e}_2) \\ : \ |k_1+\texttt{e}_1\texttt{<1>}-\texttt{e}_2\texttt{<2>}| \le k_2 \\ \qquad \Longrightarrow \texttt{x1<1>}+k_1\texttt{=x<2>} \end{array}}$$

Above Threshold

```
g(D)=max_{i<k} q_i(D)+r_i
```

$$\Pr_{x \sim AT(D)}[x = k \mid r_{-k}] = \Pr_{nT, r_k}[nT > g(D) \wedge q_k(D) + r_k > nT \mid r_{-k}]$$

$$= \Pr_{nT, r_k}[nT \in (g(D), q_k(D) + r_k) \mid r_{-k}]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

$$\leq exp(\frac{\epsilon}{2} * 1 + \frac{\epsilon}{4} * 2) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \mid r_{-k}]$$

$$= exp(\epsilon) \Pr_{nT', r'_k}[nT \in (g(D'), q_k(D') + r_k) \mid r_{-k}] = exp(\epsilon) \Pr_{x \sim AT(D')}[x = k \mid r_{-k}]$$

Above Threshold

```
forall k, |-(0,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
     cur = qi(db) + Lap(4/ε)
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1=k, cur2=cur1+1]
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

$\boxed{\text{Choosing } k_1=1}$

Above Threshold

```
forall k, |-(0,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
     cur = qi(db) + Lap(4/ε)
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1=k, cur2=cur1+1]
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

We can then reason by standard pRHL

Above Threshold

```
forall k, |-(ε,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1<>k]
  <[fun x => if x=k then ε else 0]>
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Case 2

Above Threshold

```
forall k, |-(ε,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1<>k]
  <[0]>
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Case 2

Above Threshold

```
forall k, |-(0,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1<>k]
  <[0]>
     cur = qi(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output1=k => output2=k]
```

Which rule shall we apply?

apRHL
Generalized Laplace

$$\frac{}{\vdash_{k2*\varepsilon,0} \begin{array}{l} \texttt{x}_1\texttt{:=\$ Lap}(\varepsilon,\texttt{e}_1) \\ \sim \\ \texttt{x}_2\texttt{:=\$ Lap}(\varepsilon,\texttt{e}_2) \\ : |k_1+\texttt{e}_1\texttt{<1>}-\texttt{e}_2\texttt{<2>}| \le k_2 \\ \qquad \texttt{==> x1<1>+}k_1\texttt{=x<2>} \end{array}}$$

Above Threshold

$\boxed{g(D)=\max_{i<k}\ q_i(D)+r_i}$

$$\Pr_{x\sim AT(D)}[x=k\,|\,r_{-k}] = \Pr_{nT,r_k}[nT > g(D) \wedge q_k(D)+r_k > nT\,|\,r_{-k}]$$

$$= \Pr_{nT,r_k}[nT \in (g(D), q_k(D)+r_k)\,|\,r_{-k}]$$

Now let's define: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

$$nT' = nT + g(D) - g(D')$$

$$\leq exp(\frac{\epsilon}{2} * 1 + \frac{\epsilon}{4} * 2) \Pr_{nT',r'_k}[nT \in (g(D'), q_k(D')+r_k)\,|\,r_{-k}]$$

$$= exp(\epsilon) \Pr_{nT',r'_k}[nT \in (g(D'), q_k(D')+r_k)\,|\,r_{-k}] = exp(\epsilon) \Pr_{x\sim AT(D')}[x=k\,|\,r_{-k}]$$

Above Threshold

```
forall k, |-(0,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1<>k,cur2≤cur1+1]
  <[0]>
      if (cur ≥ T /\ output = N )
         output = i;
      i++
  return output;
[output1=k => output2=k]
```

Which rule shall we apply?

Above Threshold

```
forall k, |-(0,0)
AboveT (q1,…,qk : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
      cur = qi(db) + Lap(4/ε)
  [adj b1 b2,GS(qi)≤1,…, nT2=nT1 + 1, invariant,i1<>k,cur2≤cur1+1]
  <[0]>
      if (cur ≥ T /\ output = N )
          output = i;
      i++
  return output;
[output1=k => output2=k]
```

We can then reason by standard pRHL