CS 591: Formal Methods in Security and Privacy

Differential Privacy examples

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

Where we were…

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)

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;
```

apRHL: pointwise DP rule

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

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

$$\sum \delta_r \leq \delta$$

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

apRHL
Generalized Laplace

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

Probabilistic Relational Hoare Logic

Composition

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

Probabilistic Relational Hoare Logic

Consequence

$$\frac{\vdash_{\varepsilon,\delta} c_1 \sim c_2 : S \Rightarrow R \qquad \varepsilon \leq \varepsilon' \quad \delta \leq \delta' \quad P \Rightarrow S \quad R \Rightarrow Q}{\vdash_{\varepsilon',\delta'} c_1 \sim c_2 : P \Rightarrow Q}$$

apRHL

awhile

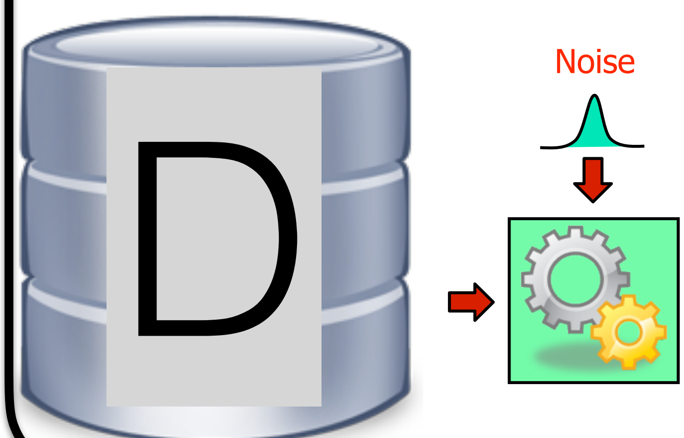
$$
\frac{\begin{array}{c} P \land e\langle 1\rangle \leq 0 \Rightarrow \neg b1\langle 1\rangle \\[1em] \vdash_{\varepsilon_k,\delta_k} c1 \sim c2 : P \land b1\langle 1\rangle \land b2\langle 2\rangle \land k = e\langle 1\rangle \land e\langle 1\rangle \leq n \\ \Longrightarrow P \land b1\langle 1\rangle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

Today: one last example of differentially private programs

Sparse Vector

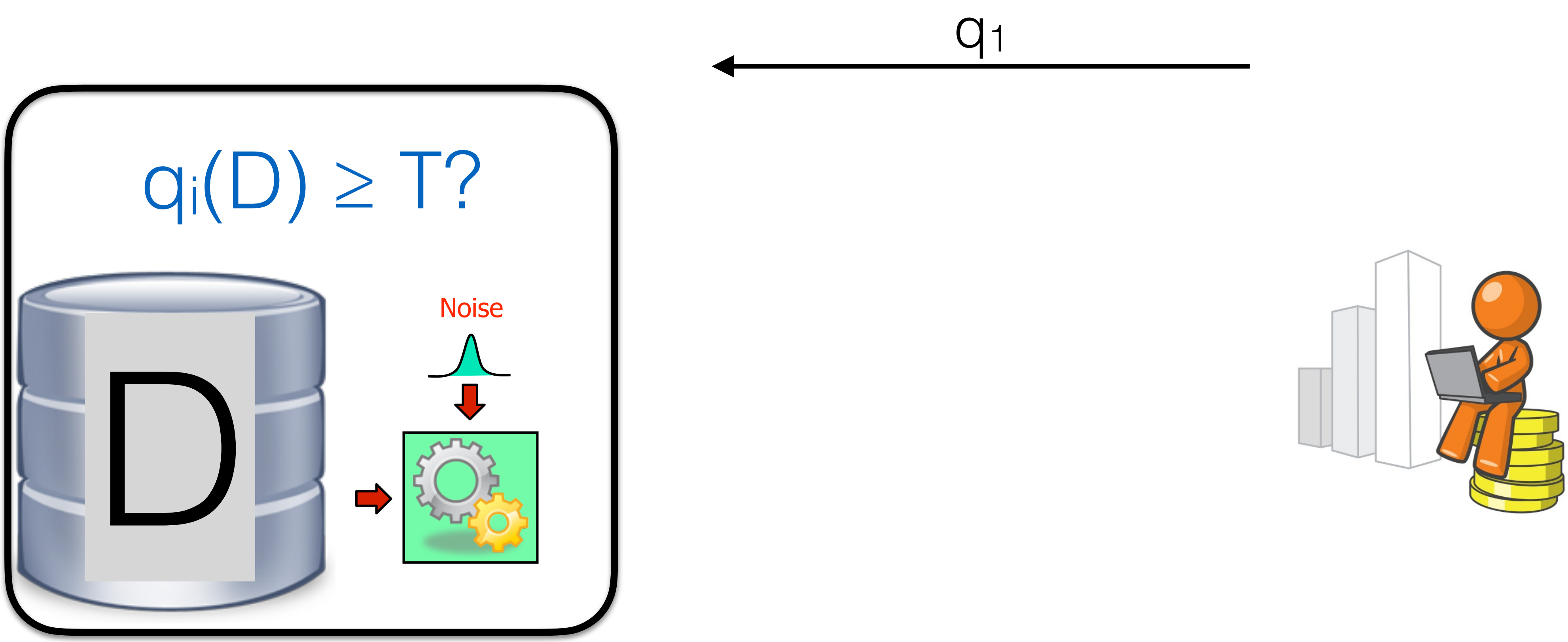
SparseVector($D, q_1, \ldots, q_n, T, \varepsilon$)

$q_i(D) \geq T?$



Sparse Vector

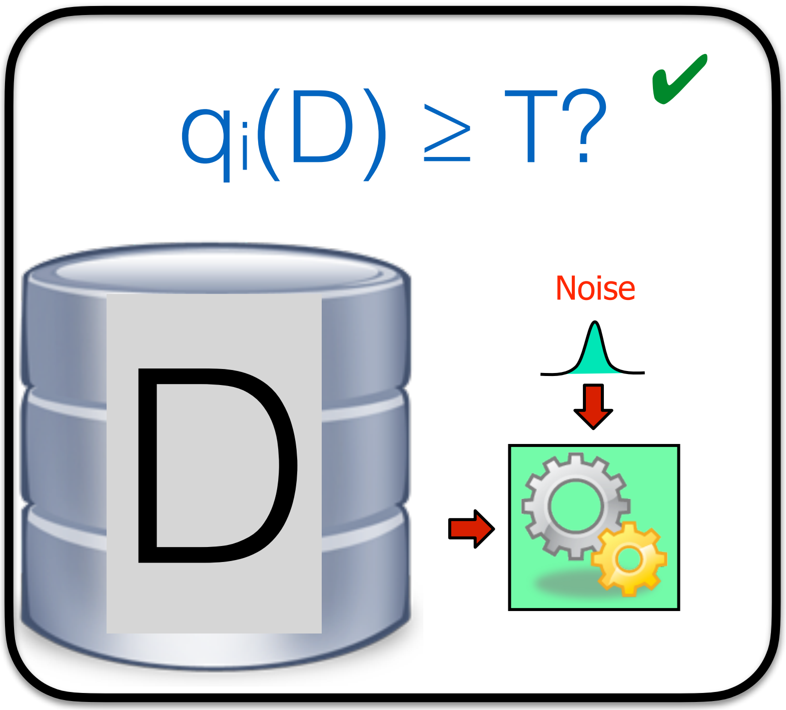
SparseVector($D, q_1, \ldots, q_n, T, \varepsilon$)



Sparse Vector

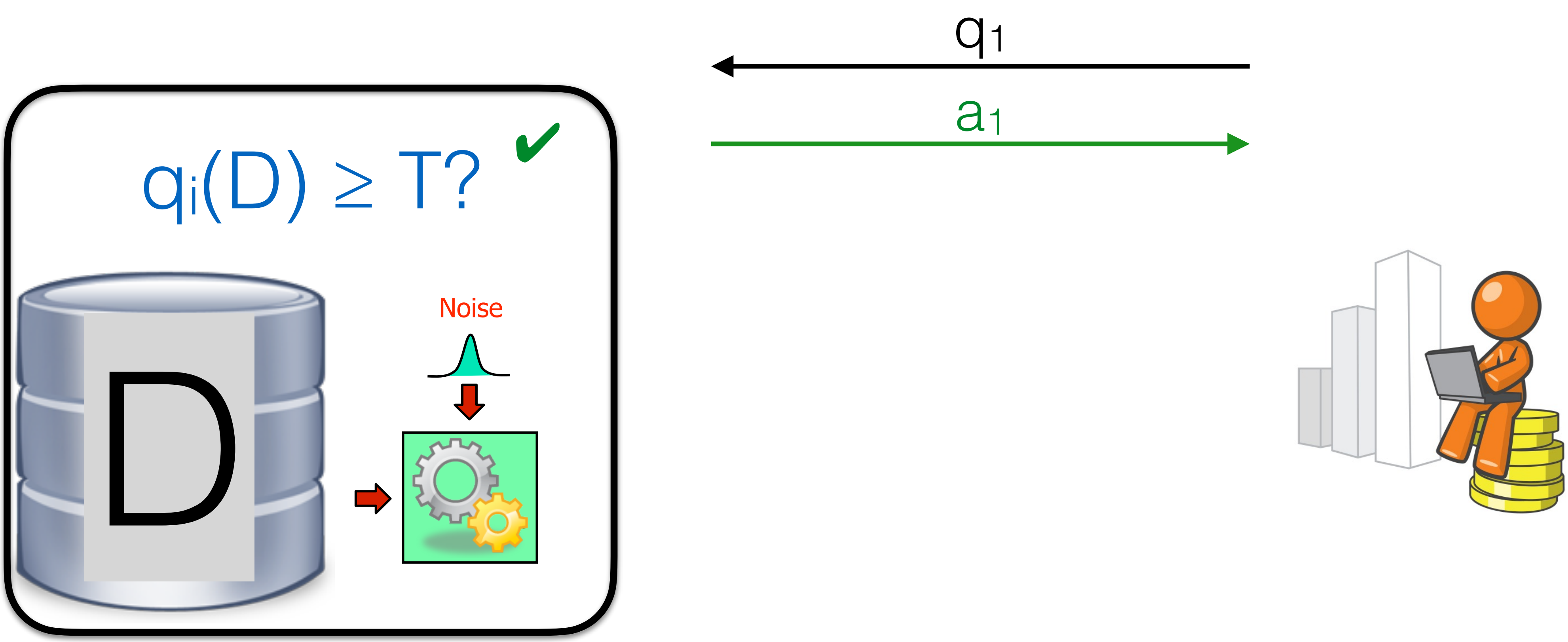
$$\text{SparseVector}(D, q_1, \ldots, q_n, T, \varepsilon)$$

q_1



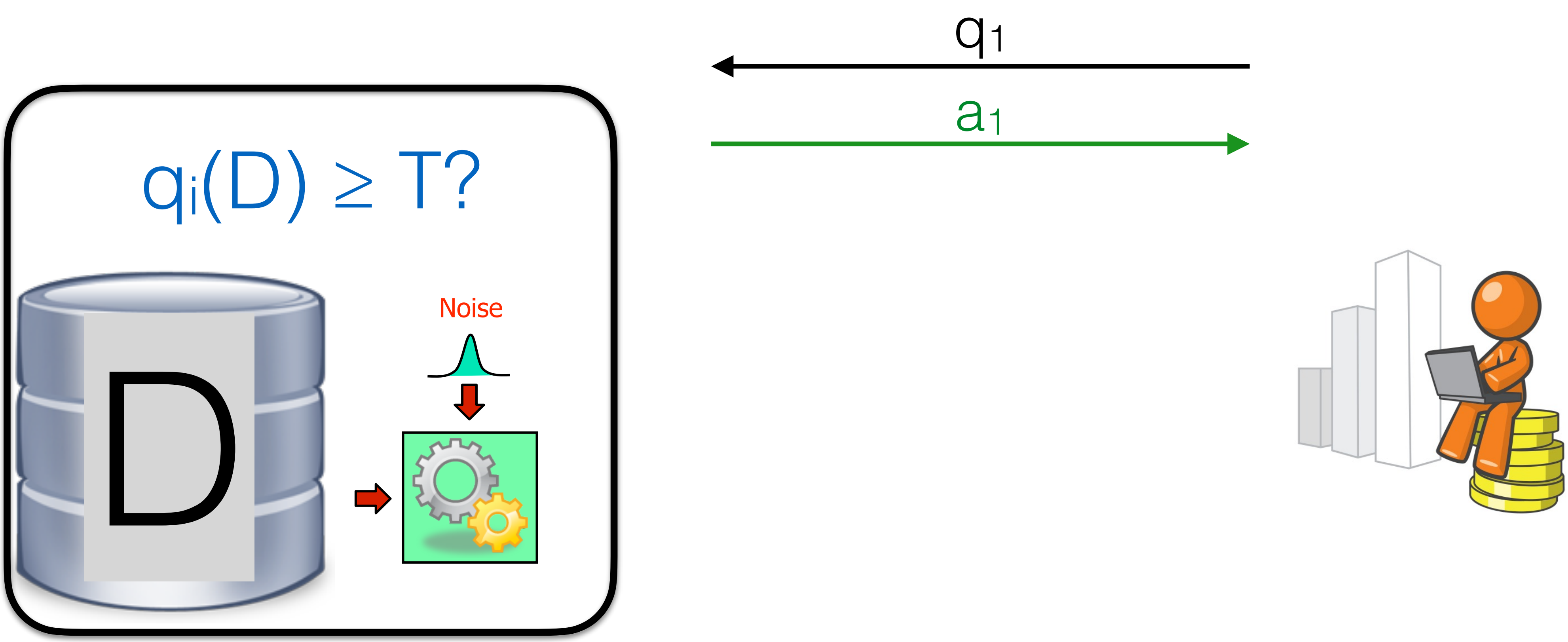
Sparse Vector

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



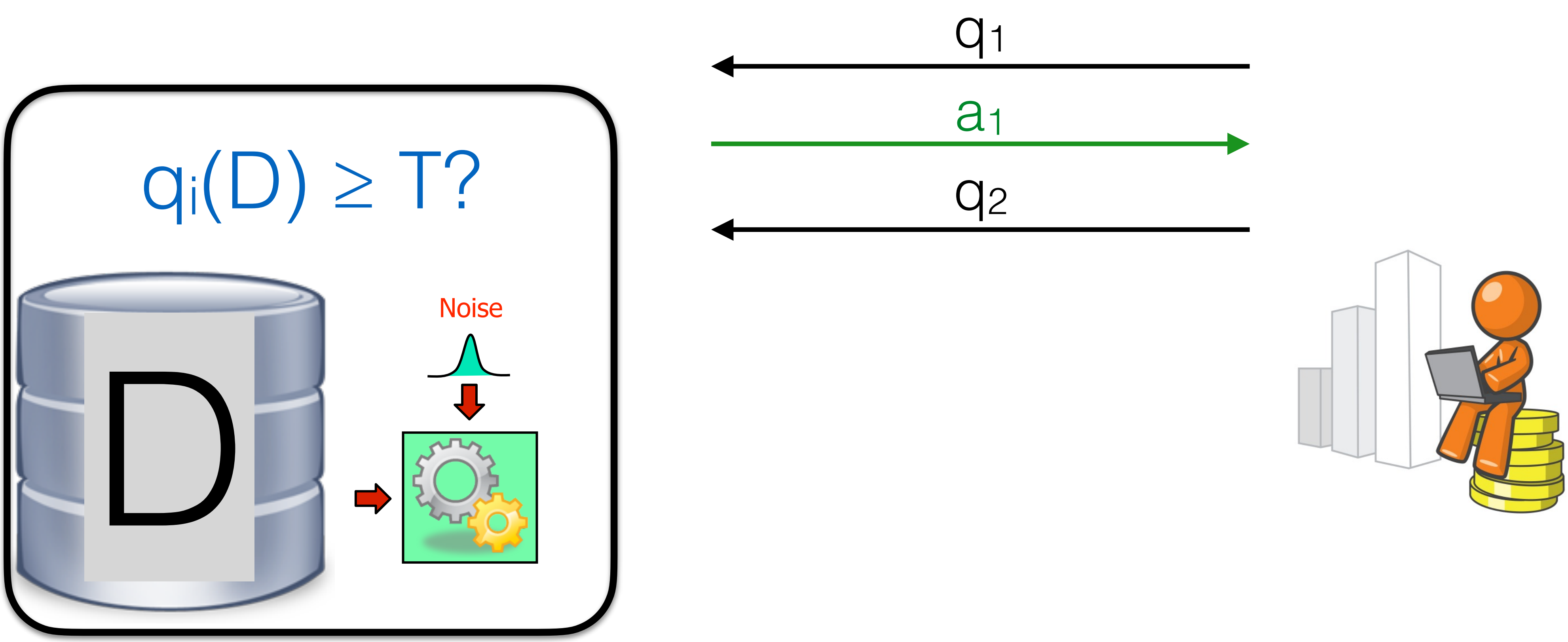
Sparse Vector

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



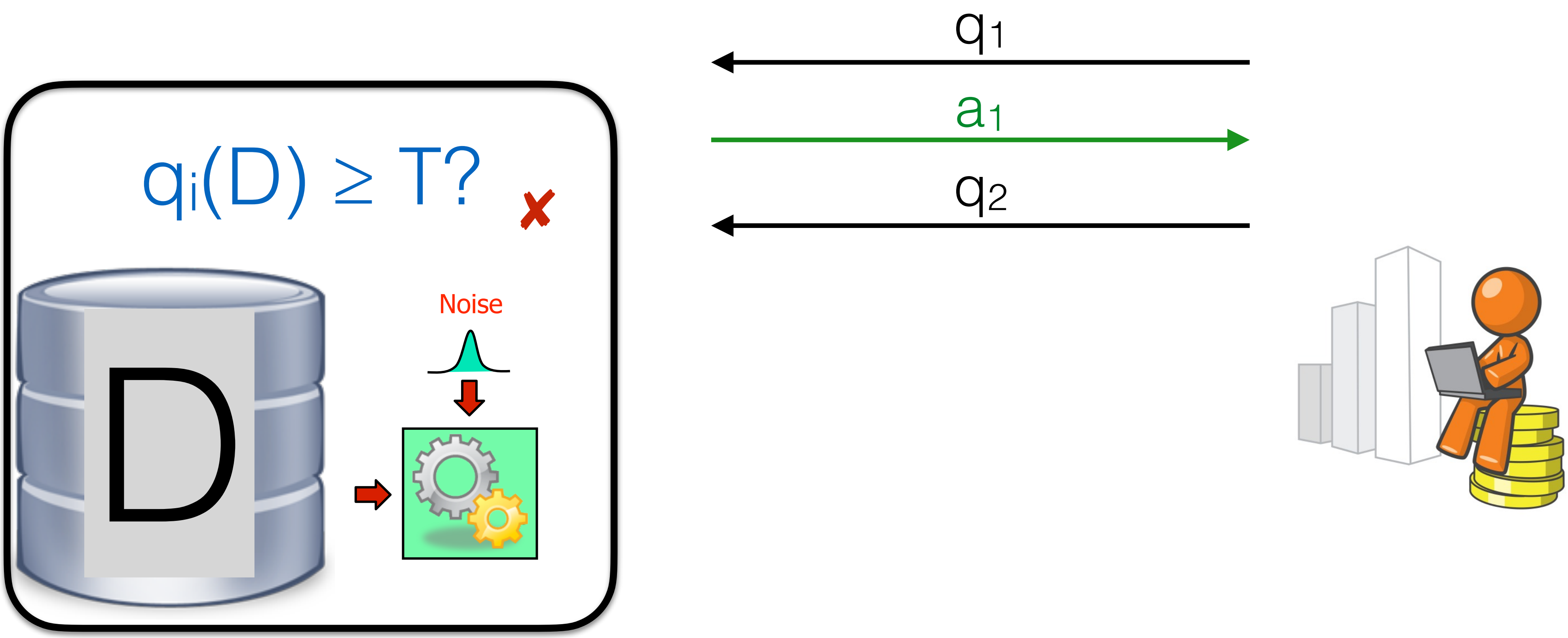
Sparse Vector

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



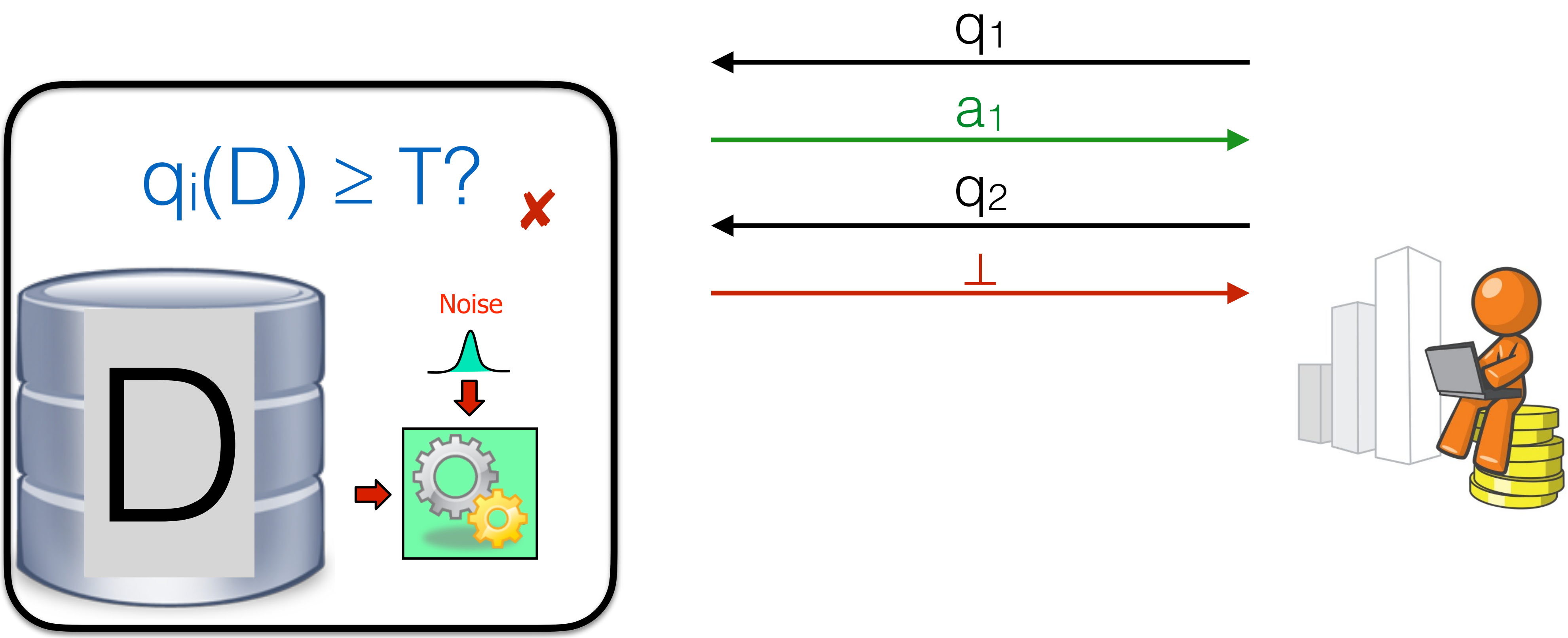
Sparse Vector

SparseVector(D,q_1,...,q_n,T,ε)



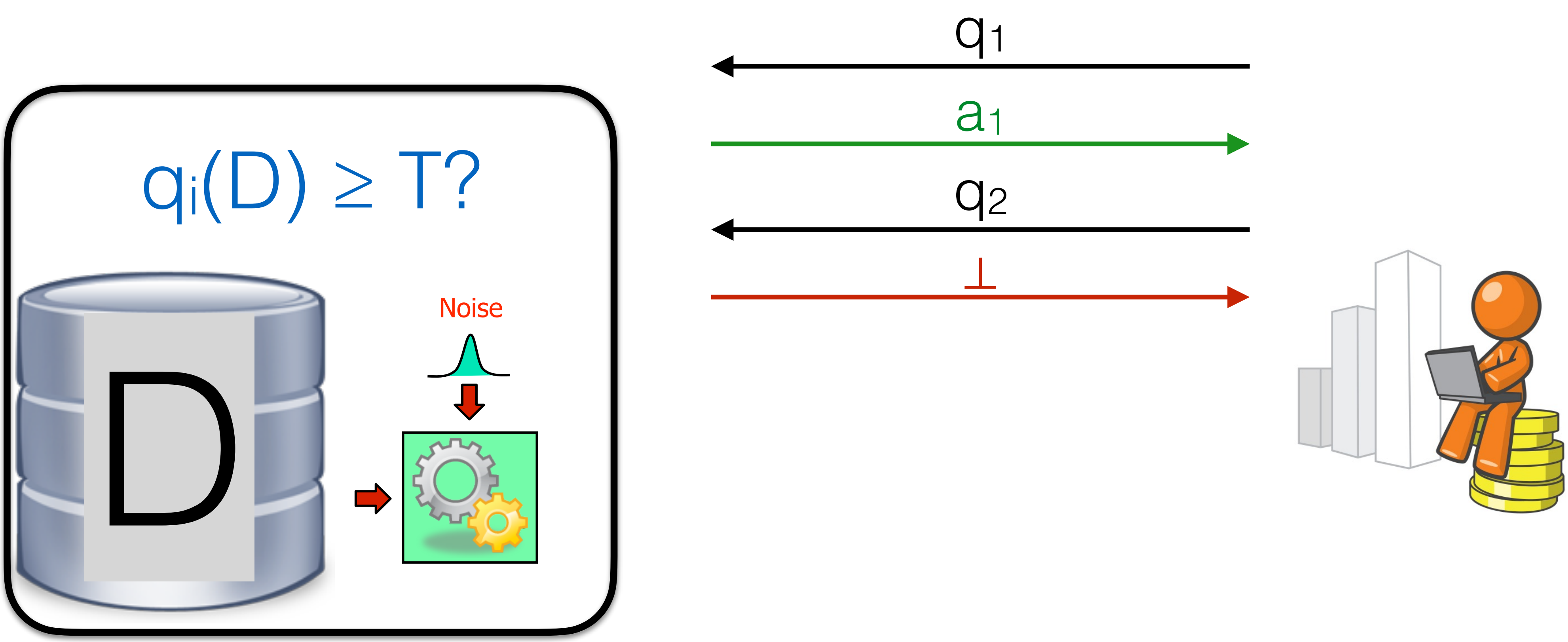
Sparse Vector

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



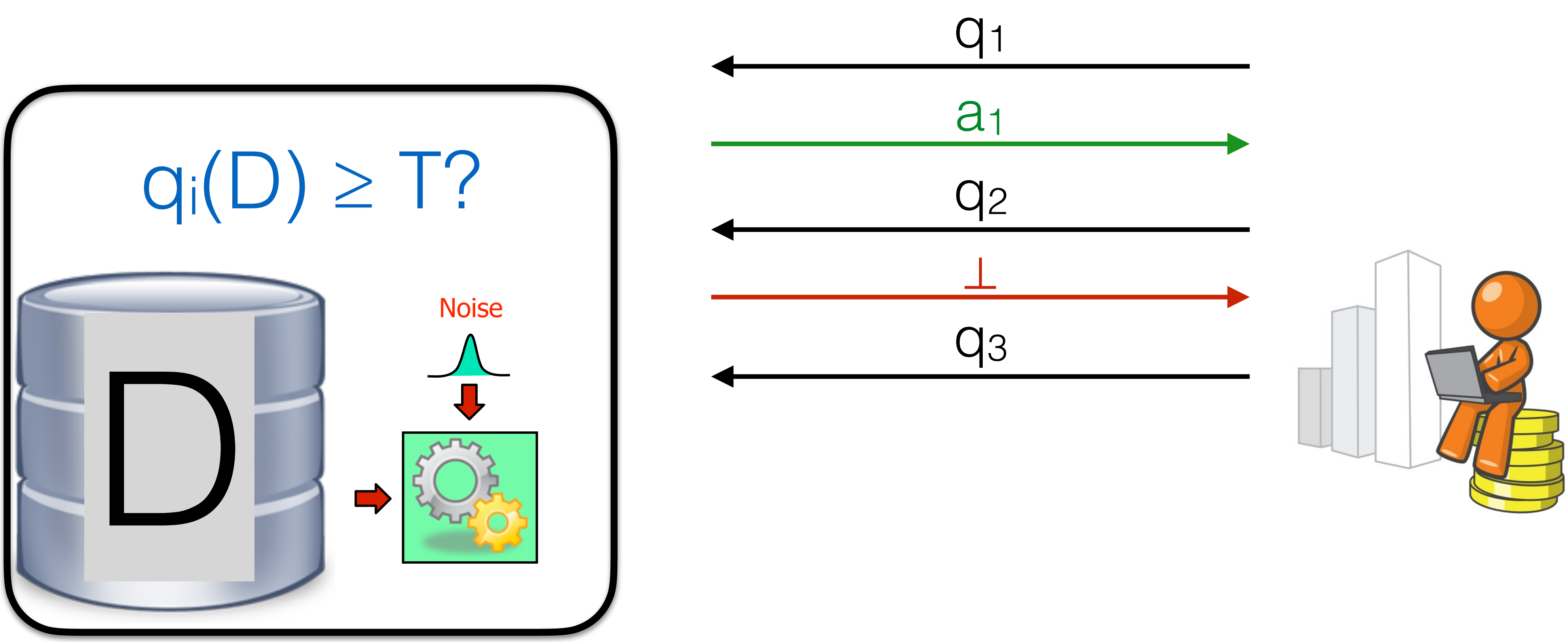
Sparse Vector

SparseVector(D, q_1, ..., q_n, T, ε)



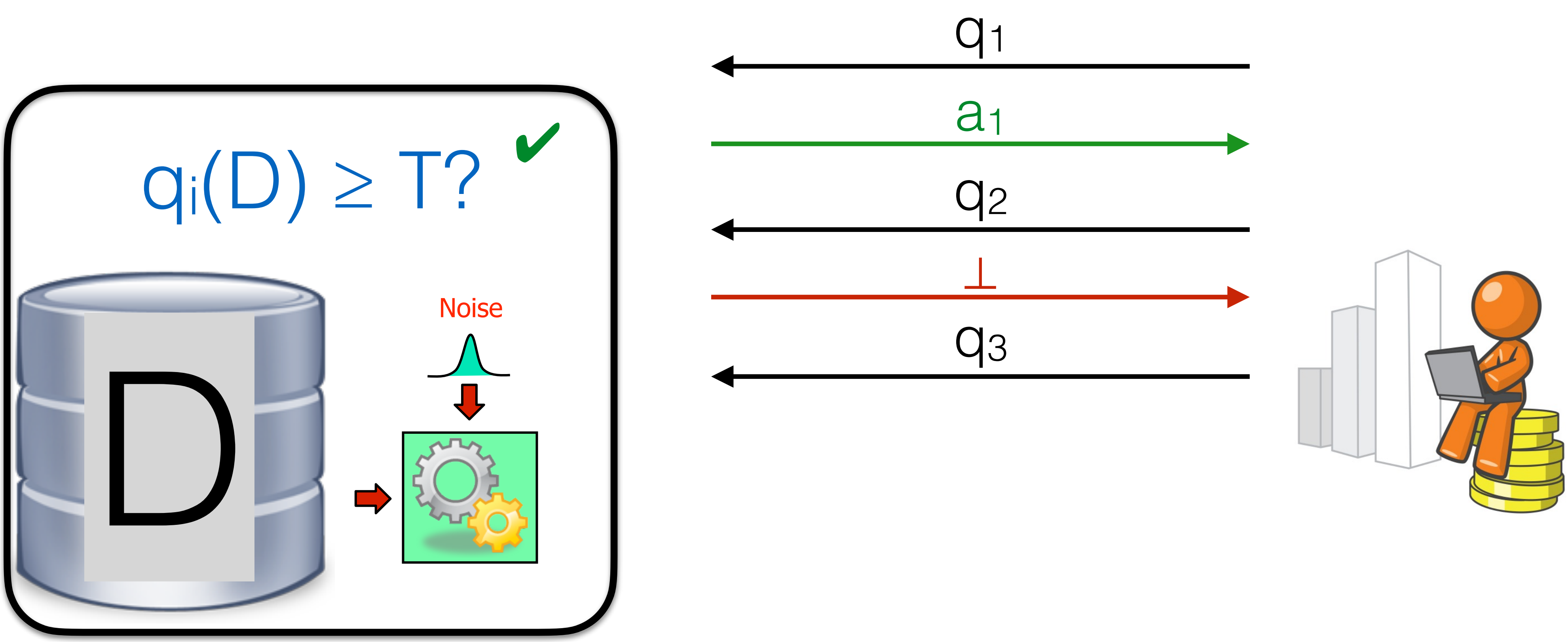
Sparse Vector

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



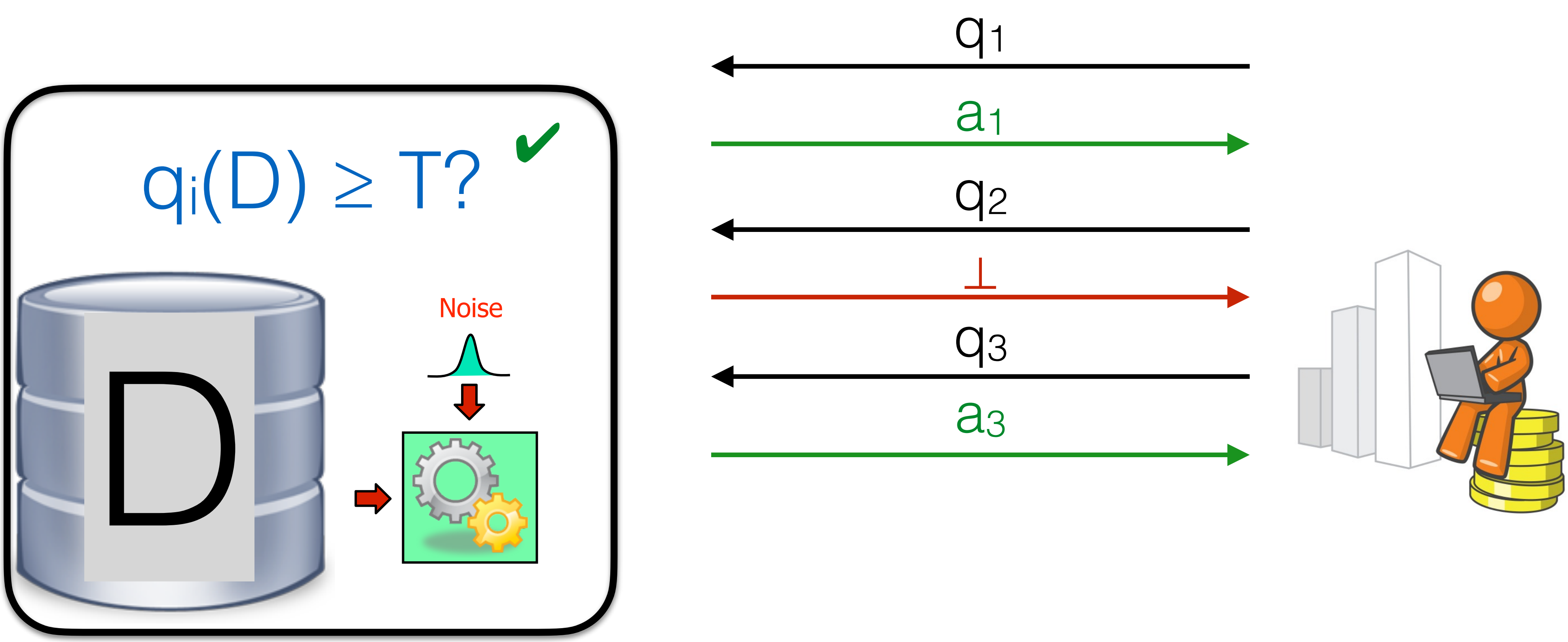
Sparse Vector

SparseVector(D,q_1,...,q_n,T,ε)



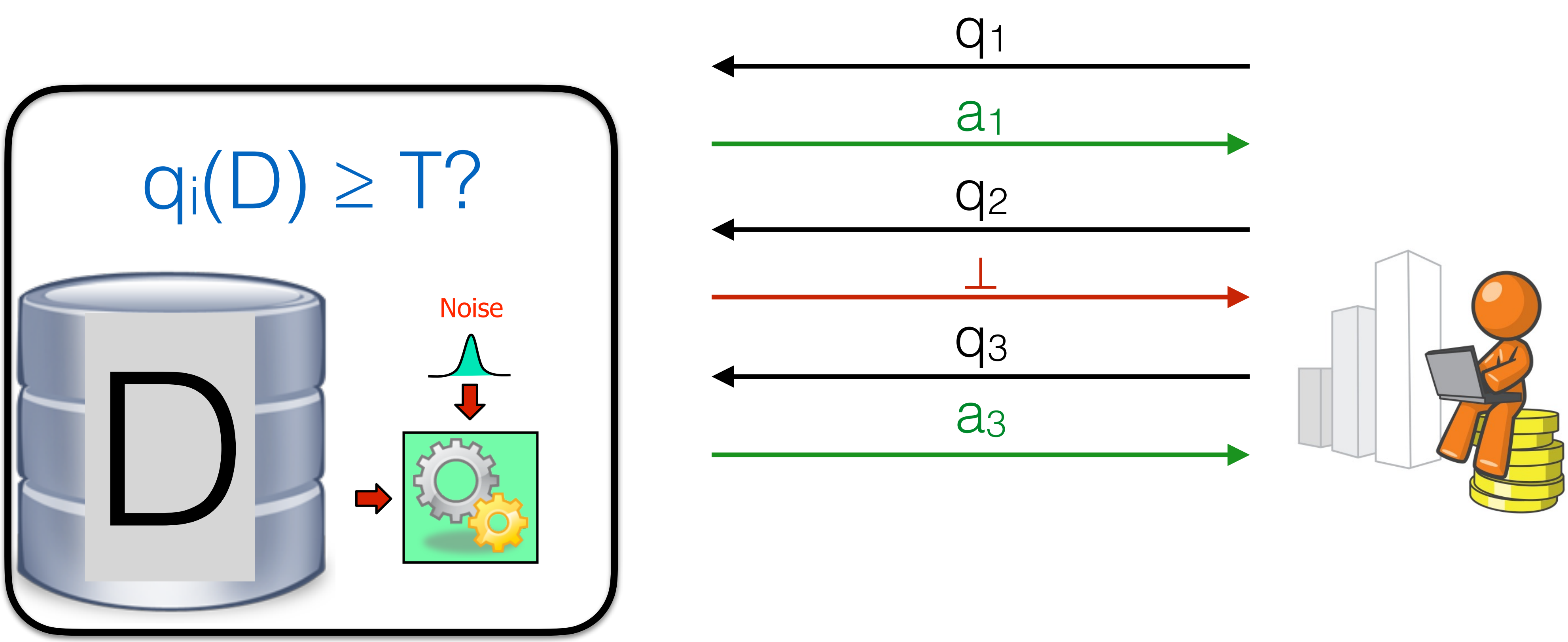
Sparse Vector

SparseVector(D,q₁,...,qₙ,T,ε)



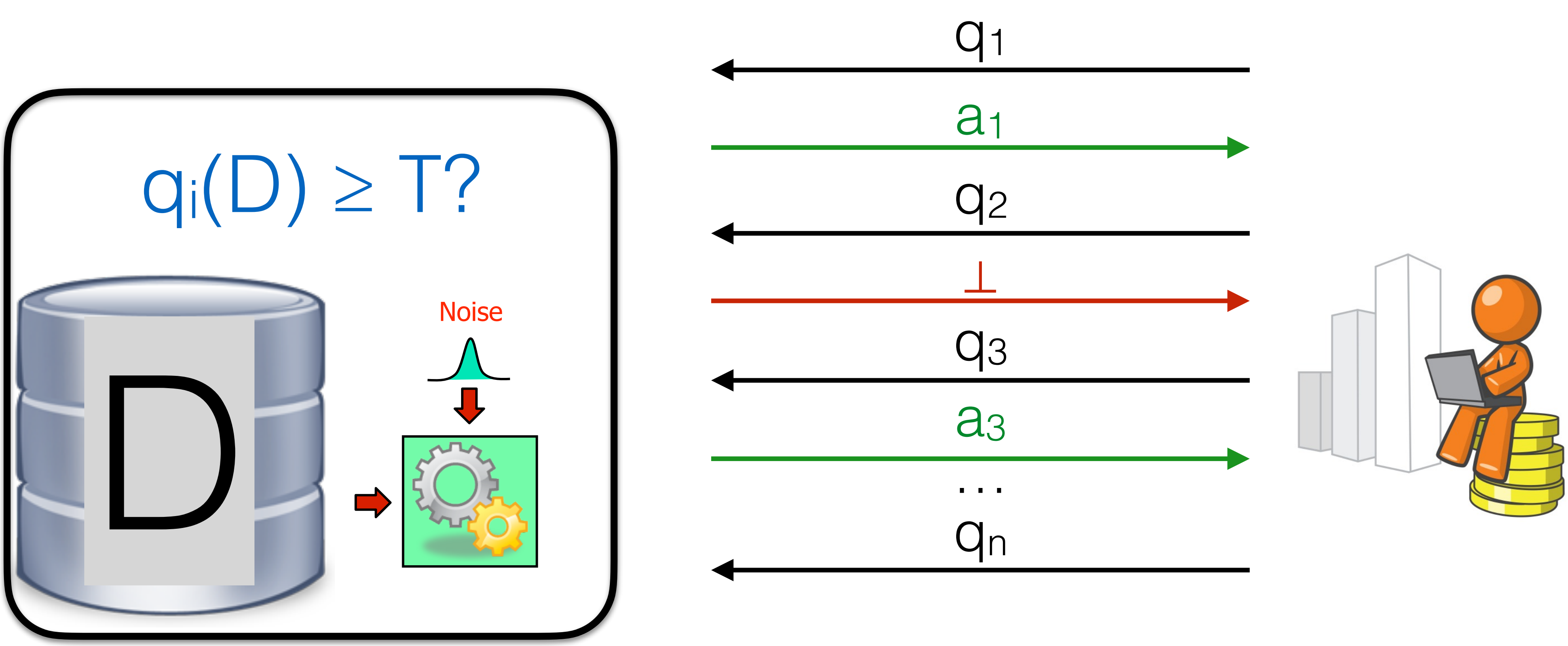
Sparse Vector

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



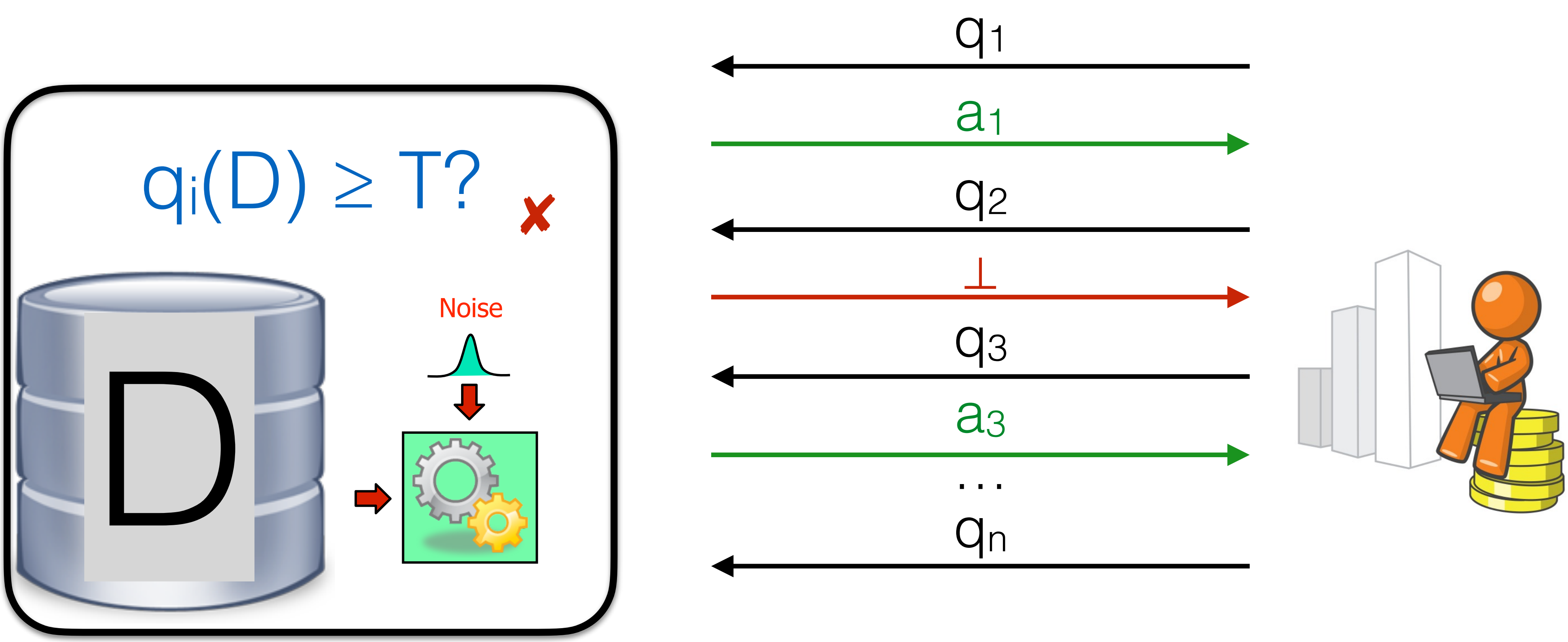
Sparse Vector

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



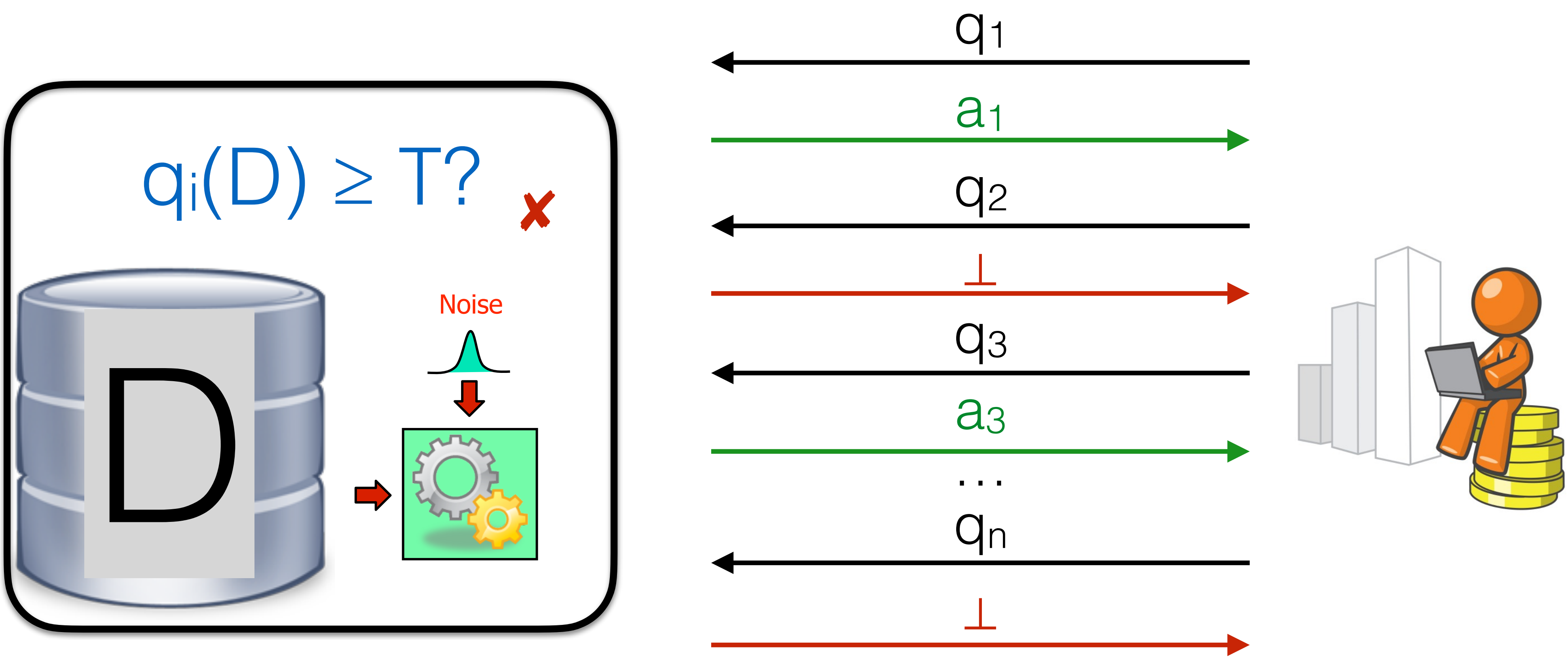
Sparse Vector

SparseVector(D,q_1,...,q_n,T,ε)



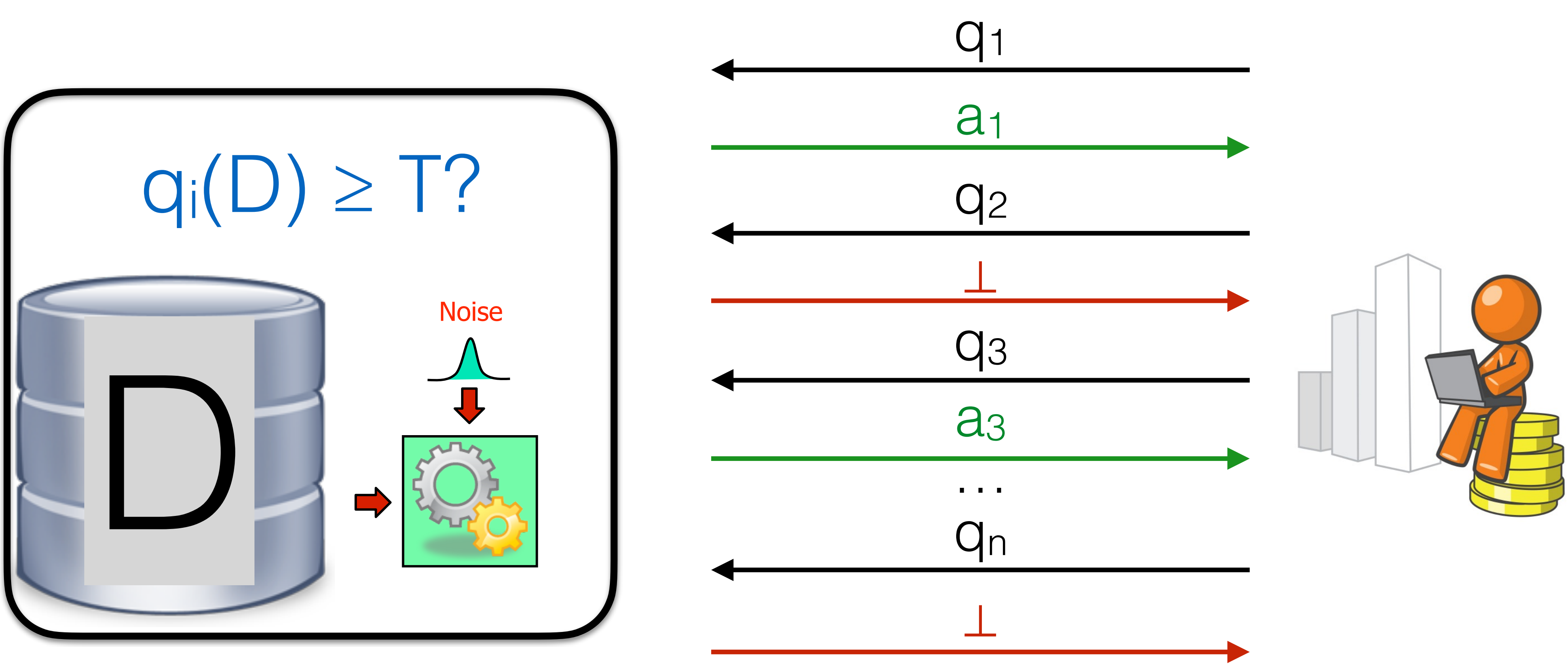
Sparse Vector

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



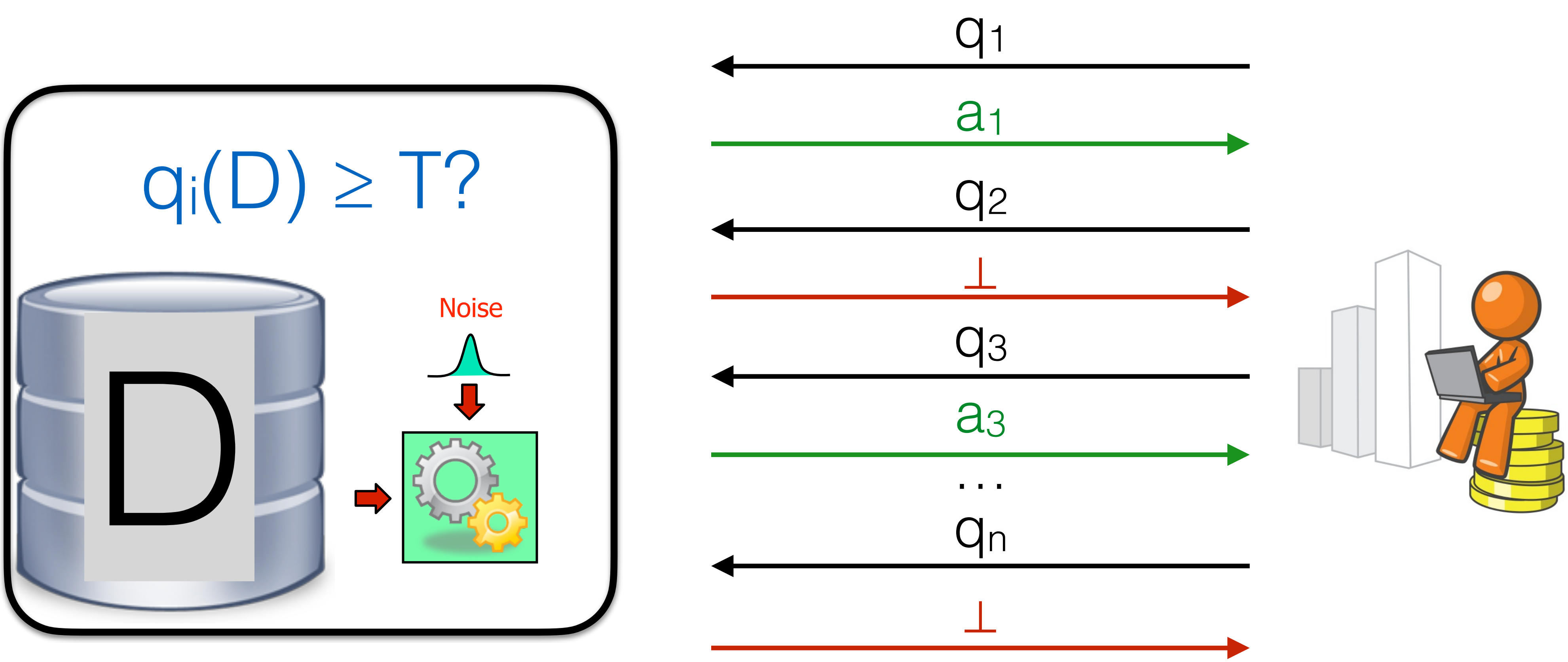
Sparse Vector

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



Sparse Vector

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



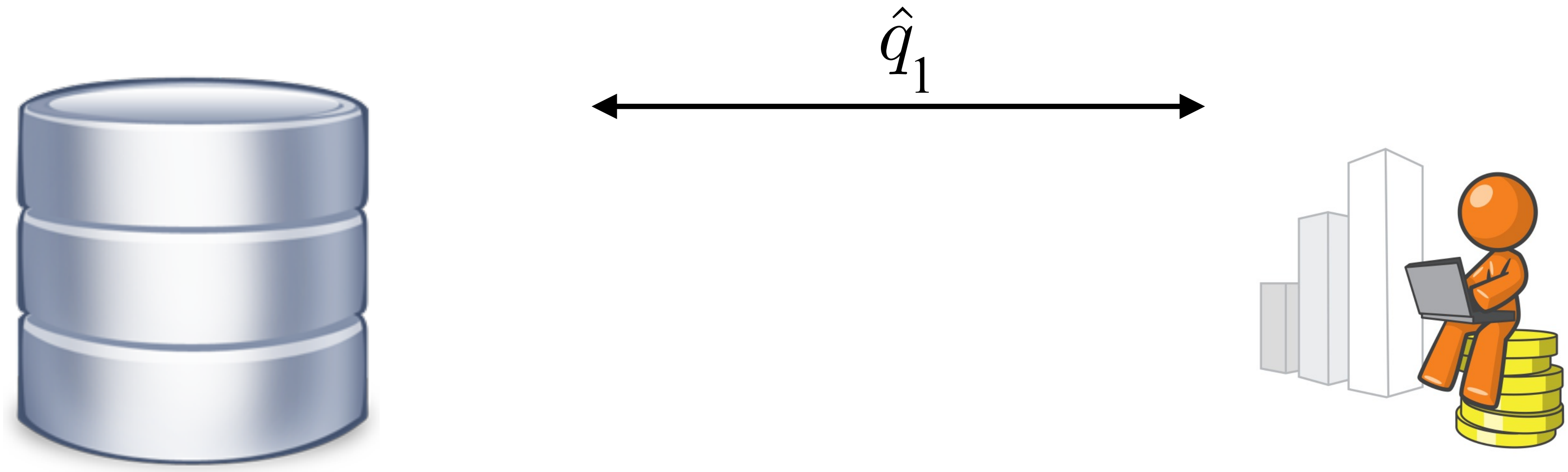
How can we achieve epsilon-DP by paying only for the queries above T?

A first step: above threshold

A first step: above threshold

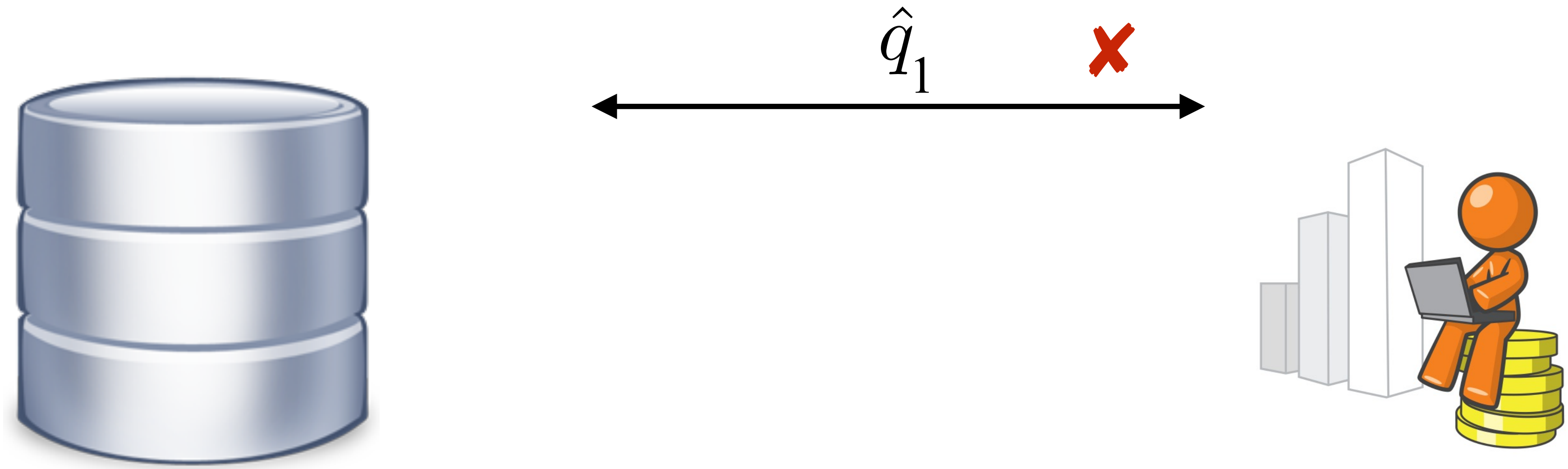
$\hat{q}_1$

A first step: above threshold



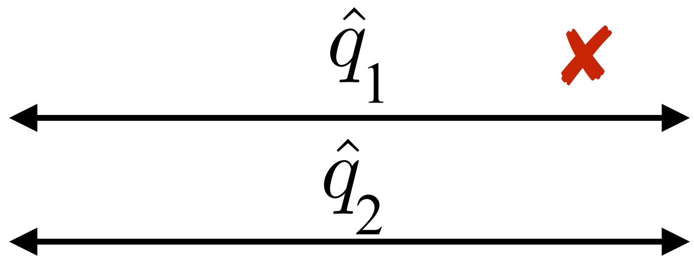
Above threshold $\hat{t}$?

A first step: above threshold



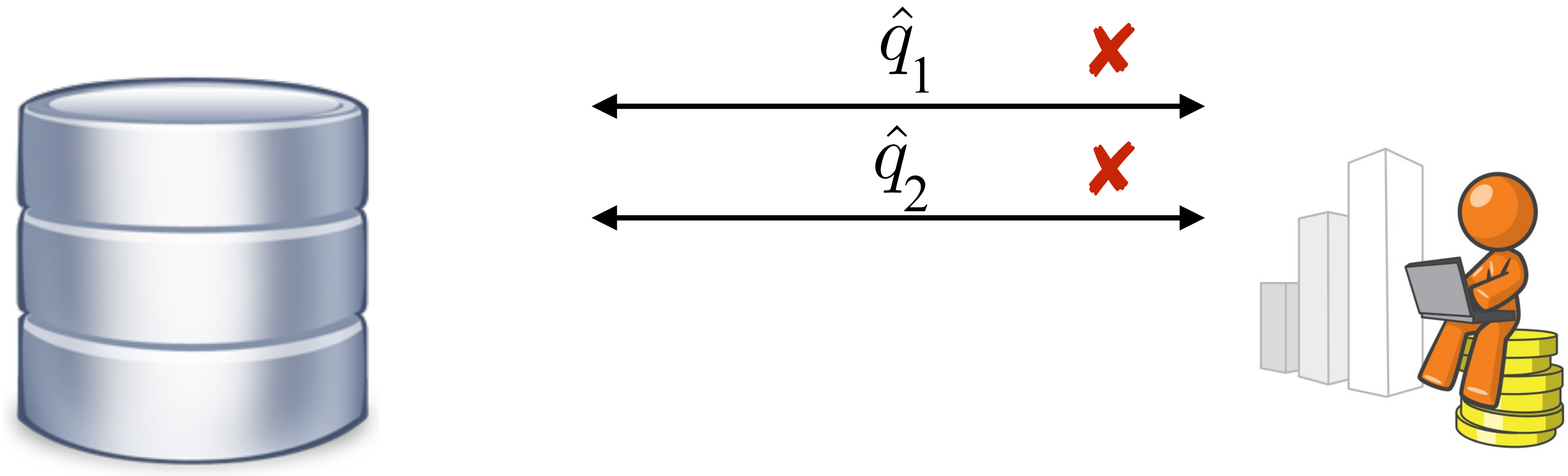
Above threshold $\hat{t}$?

A first step: above threshold



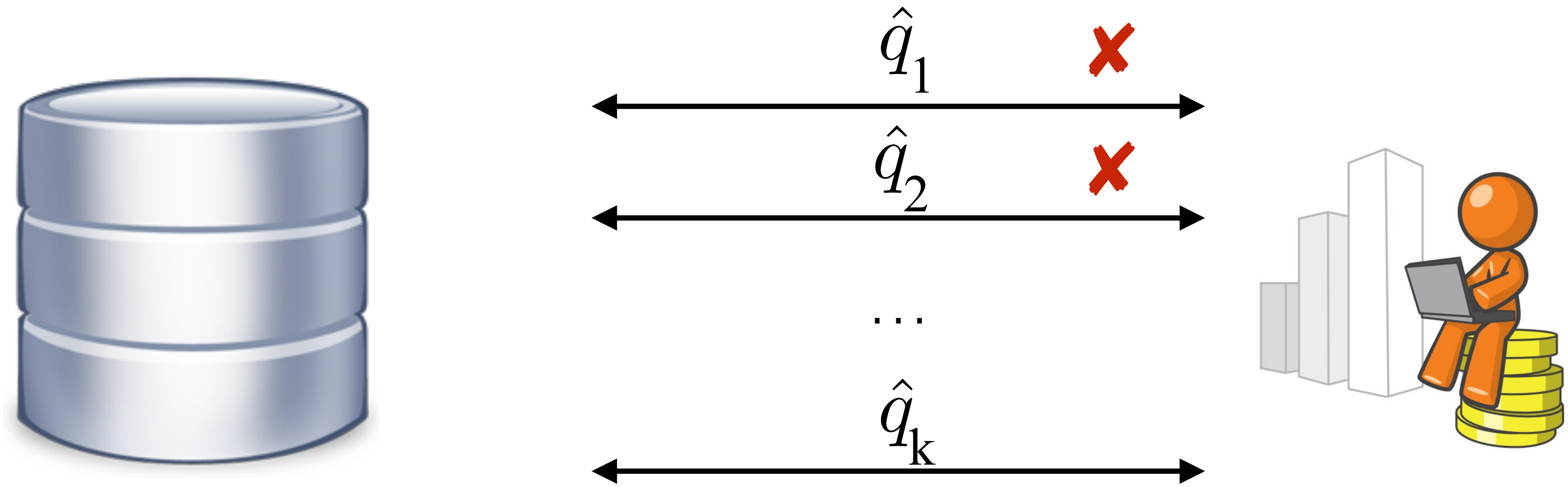
Above threshold $\hat{t}$?

A first step: above threshold



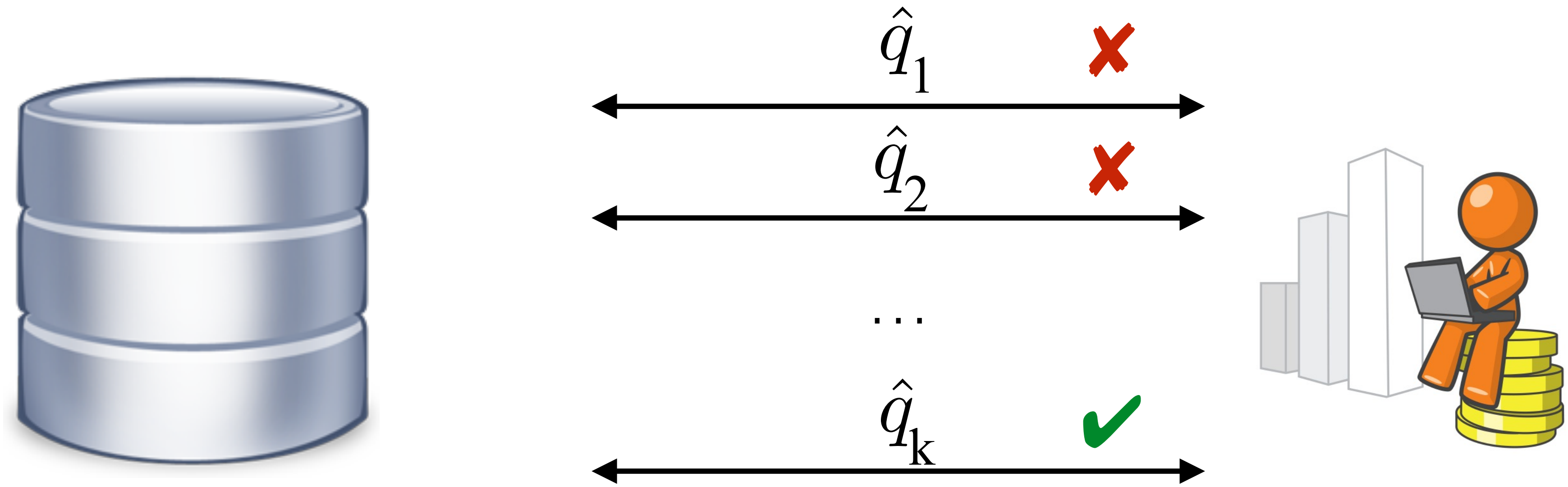
Above threshold $\hat{t}$?

A first step: above threshold



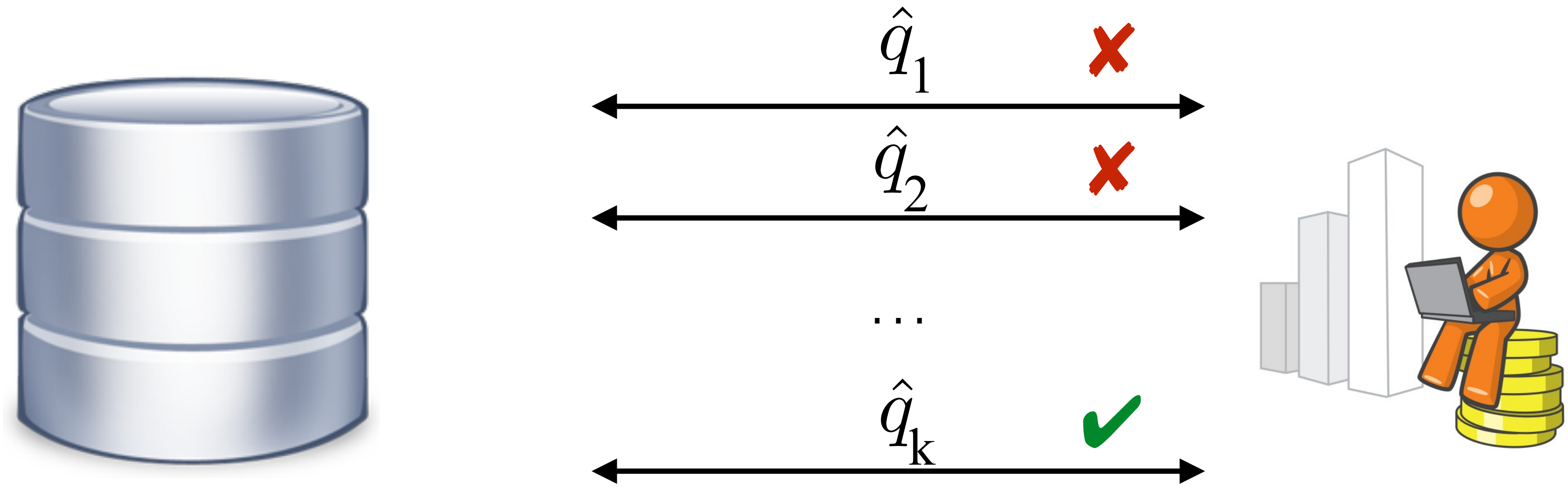
Above threshold $\hat{t}$?

A first step: above threshold



Above threshold $\hat{t}$?

A first step: above threshold



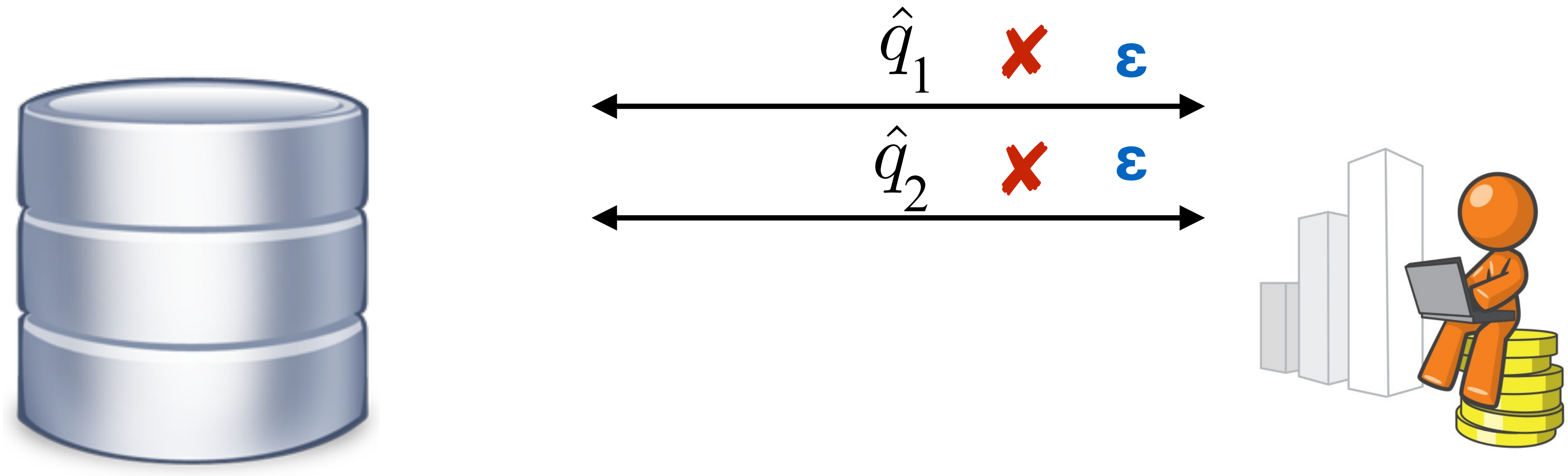
Return k

Reasoning by Composition

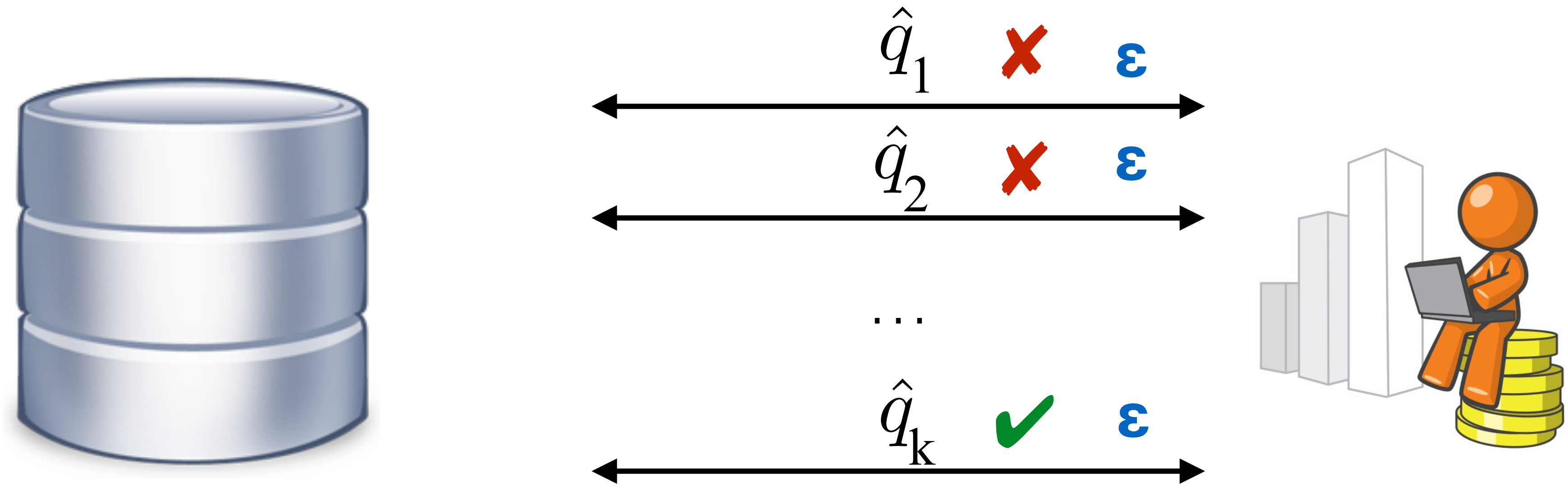
Reasoning by Composition

$\hat{q}_1$ ✘ ε

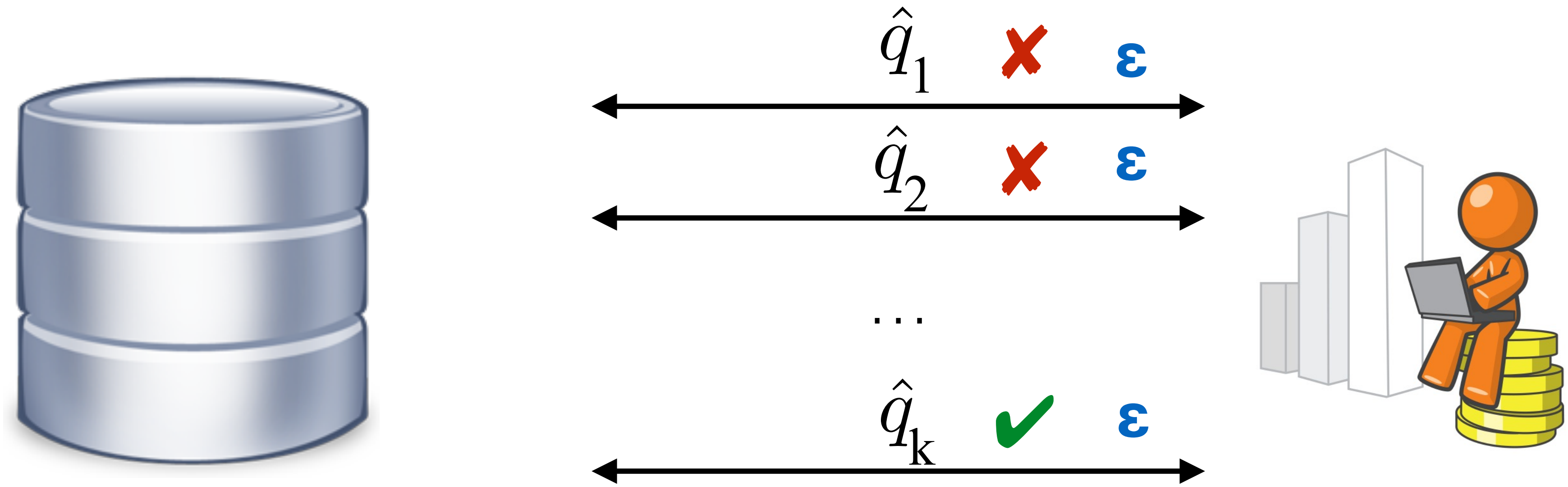
Reasoning by Composition



Reasoning by Composition

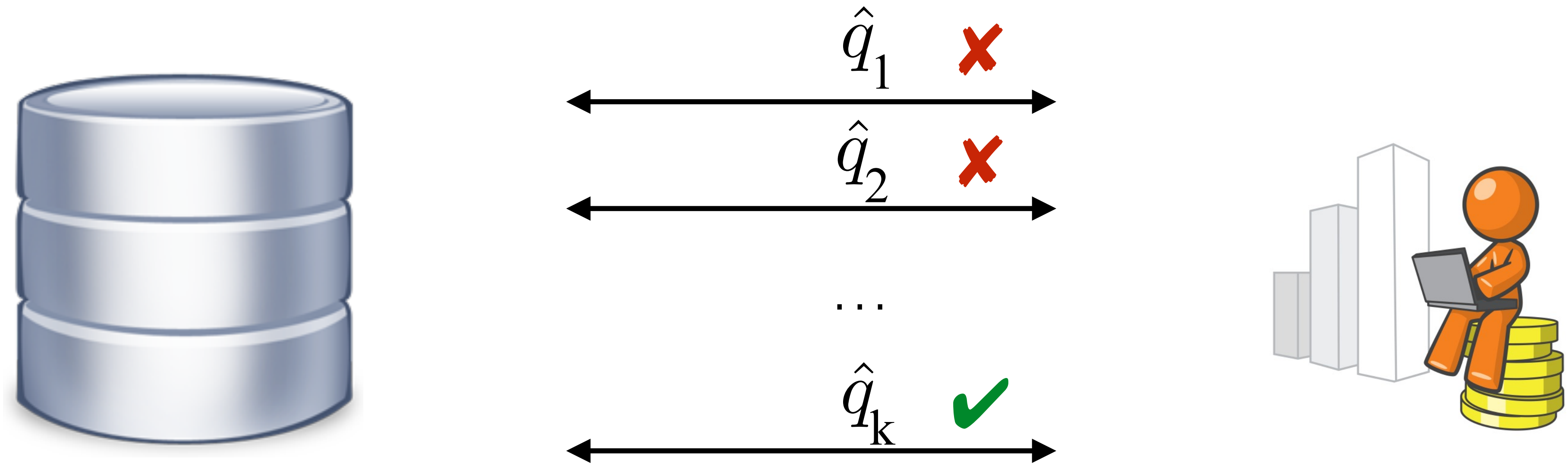


Reasoning by Composition

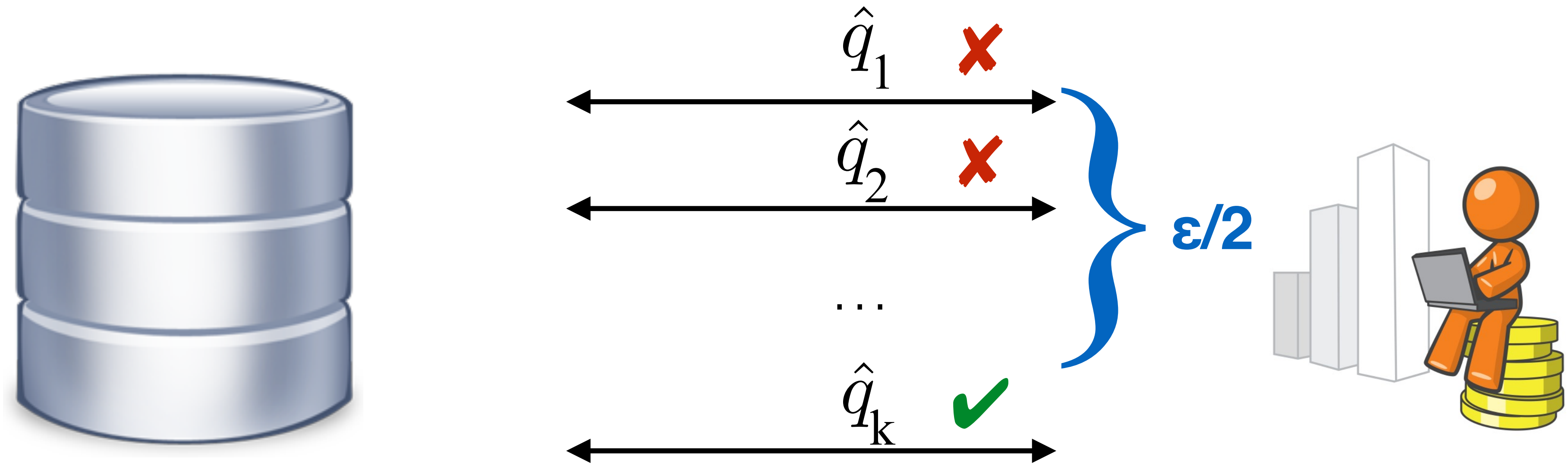


In the worst case,
the data analysis is (**nε**,0)-DP

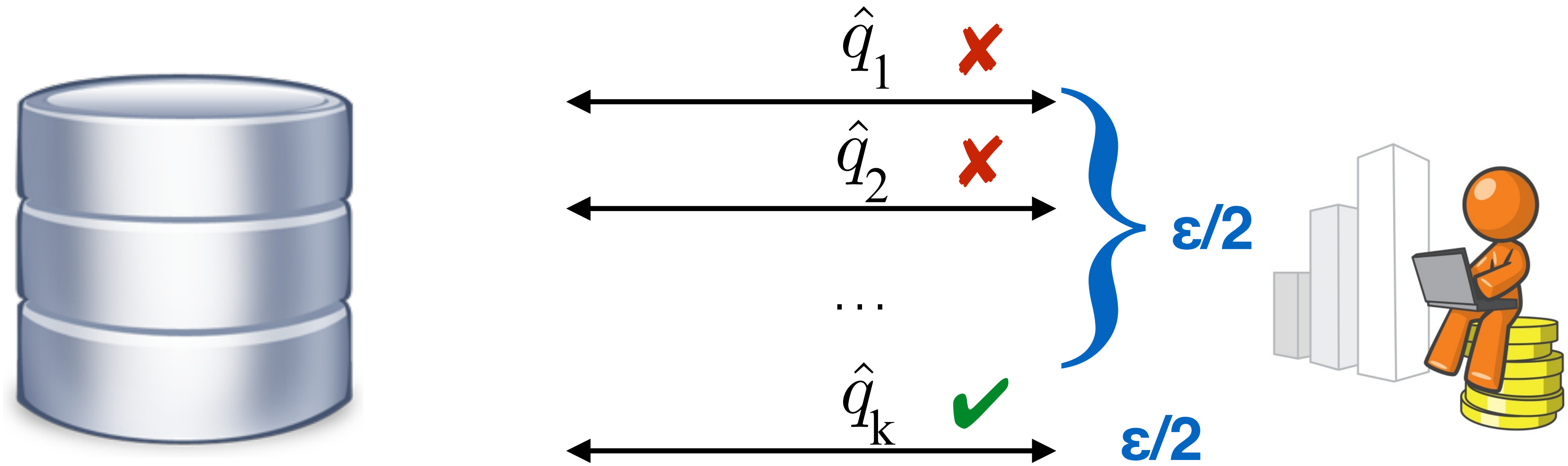
A more advanced analysis



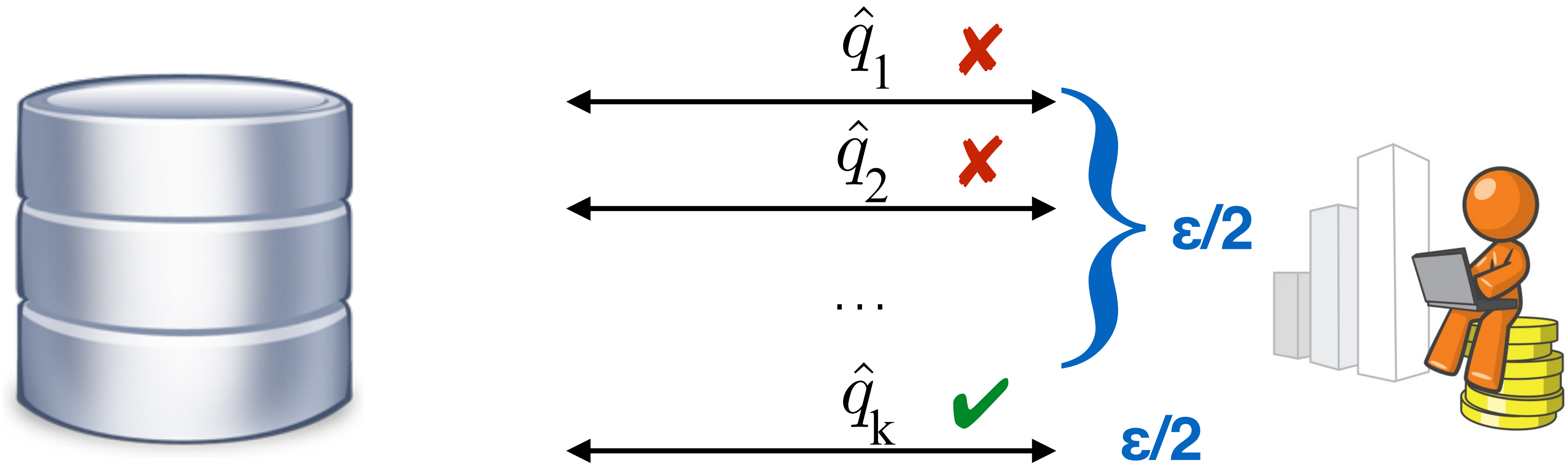
A more advanced analysis



A more advanced analysis

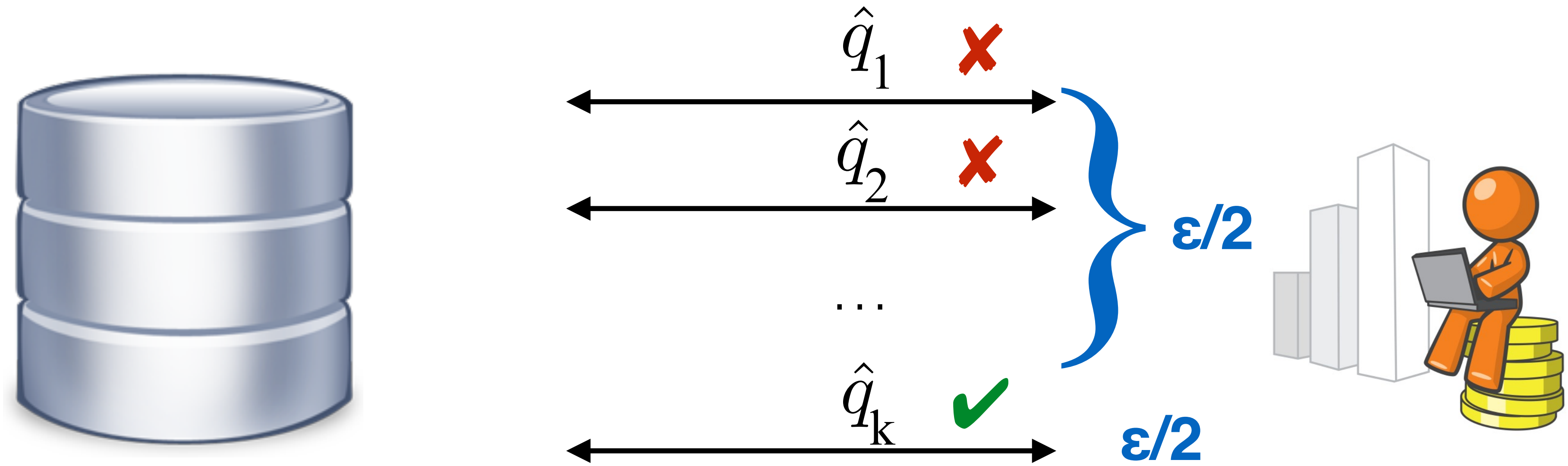


A more advanced analysis



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

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1, \quad$ 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 $a_i = \bot$

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

Input: $D, Q, \Delta, T, c.$

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(c\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1, \quad$ 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 $a_i = \bot$

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

Input: $D, Q, \Delta, T, c.$

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1),$
$\epsilon_2 = \epsilon - \epsilon_1, \quad$ 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 $a_i = \bot$

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

Input: $D, Q, \Delta, T, c.$

$\epsilon_1 = \epsilon/4, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1, \quad$ 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 $a_i = \bot$

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

Input: $D, Q, \Delta, T.$

$\epsilon_1 = \epsilon/2, \quad \rho = \mathsf{Lap}\,(\Delta/\epsilon_1)$
$\epsilon_2 = \epsilon - \epsilon_1$
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 $a_i = \bot$

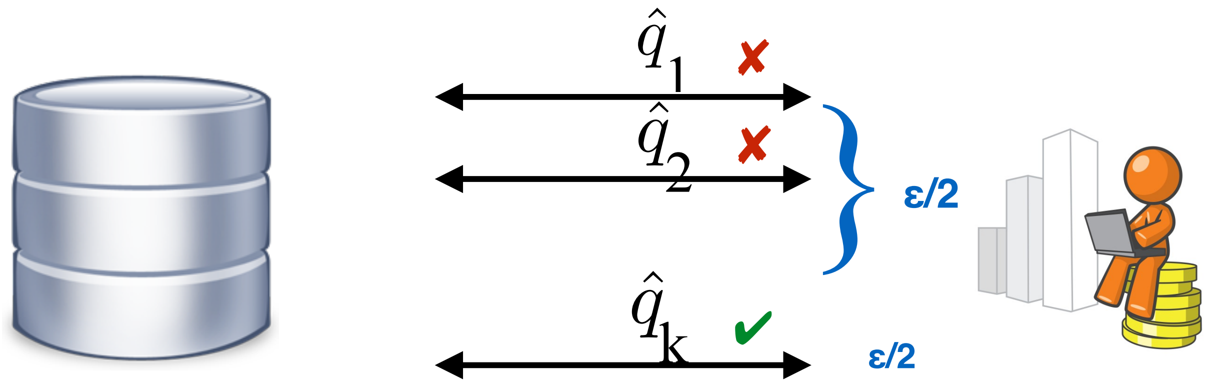
Algorithm 6 SVT in Chen et al. 2015 [1].

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 $a_i = \bot$

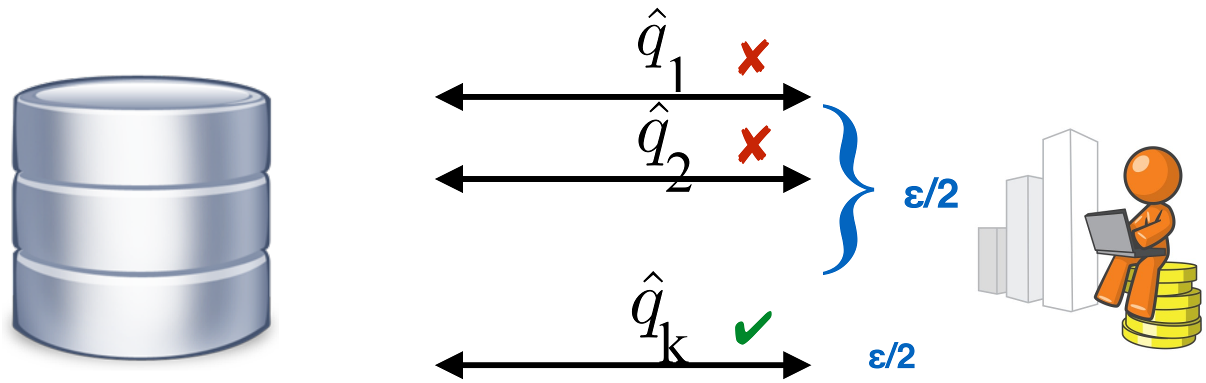
Min Lyu, Dong Su, Ninghui Li:
Understanding the Sparse Vector Technique for Differential Privacy. PVLDB (2017)

Above Threshold



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

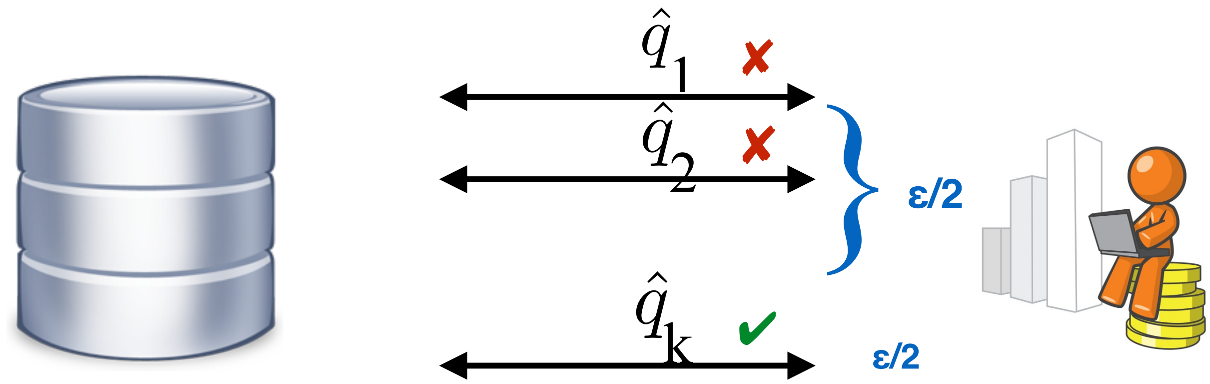
Above Threshold



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

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

Above Threshold



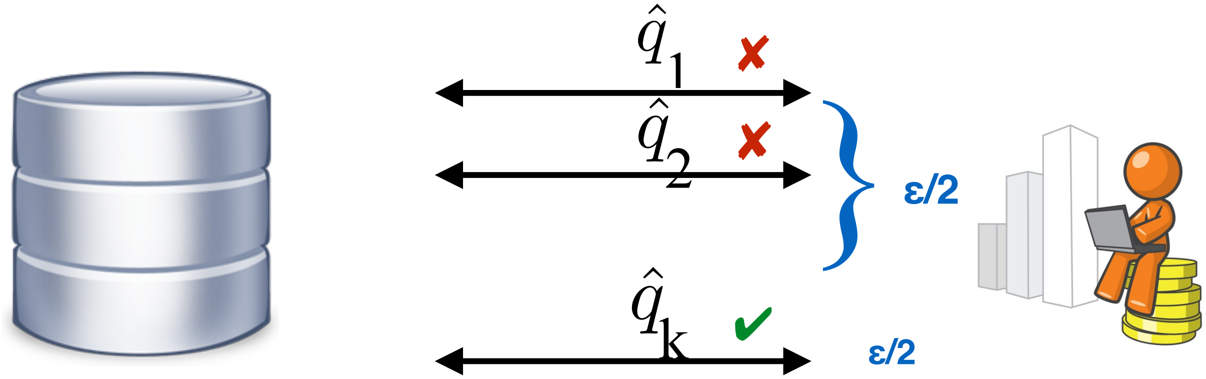
Notation $\mathtt{r_j}$ noise added at round `j`.

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

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

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

We want to show:

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

By fixing the noises $\mathtt{r}_\mathtt{j}$ for all $\mathtt{j} \leq \mathtt{k}$ we can compute the following quantity

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

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

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

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

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

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

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

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

Above Threshold

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

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

Above Threshold

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

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

Above Threshold

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

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

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

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: $r_k' = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

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

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: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

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

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

Above Threshold

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

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

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

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

```
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

$\vdash (\varepsilon, 0)$

$[adj\ b_1\ b_2, GS(q_i) \leq 1, \ldots]$

```
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

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

Above Threshold

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

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

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

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, |-(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 (q_1,…,q_k : list data → R,
        db : list data, T:R, ε: R) : int
  i = 1;
  output = N;
  nT = T + Lap(2/ε)
  while (i < N){
  [adj b_1 b_2,GS(q_i)≤1,…, nT_2=nT_1 + 1, invariant,i_1=k]
  <[fun x => if x=k then ε/2 else 0]>
     cur = q_i(db) + Lap(4/ε)
     if (cur ≥ T /\ output = N )
        output = i;
     i++
  return output;
[output_1=k => output_2=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]
```

$\boxed{\text{Case 1}}$

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

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

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]
     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

$$\frac{}{\vdash_{k2*\varepsilon,0} \; \texttt{x}_1 \texttt{:=\$ Lap}(\varepsilon,\texttt{e}_1) \sim \texttt{x}_2 \texttt{:=\$ Lap}(\varepsilon,\texttt{e}_2) : |k_1+e_1\langle 1\rangle - e_2\langle 2\rangle| \le k_2 \implies x1\langle 1\rangle + k_1 = x\langle 2\rangle}$$

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: $r'_k = r_k + g(D) - g(D') + q_k(D') - q_k(D)$

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

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

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

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]
  <[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