CS 591: 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

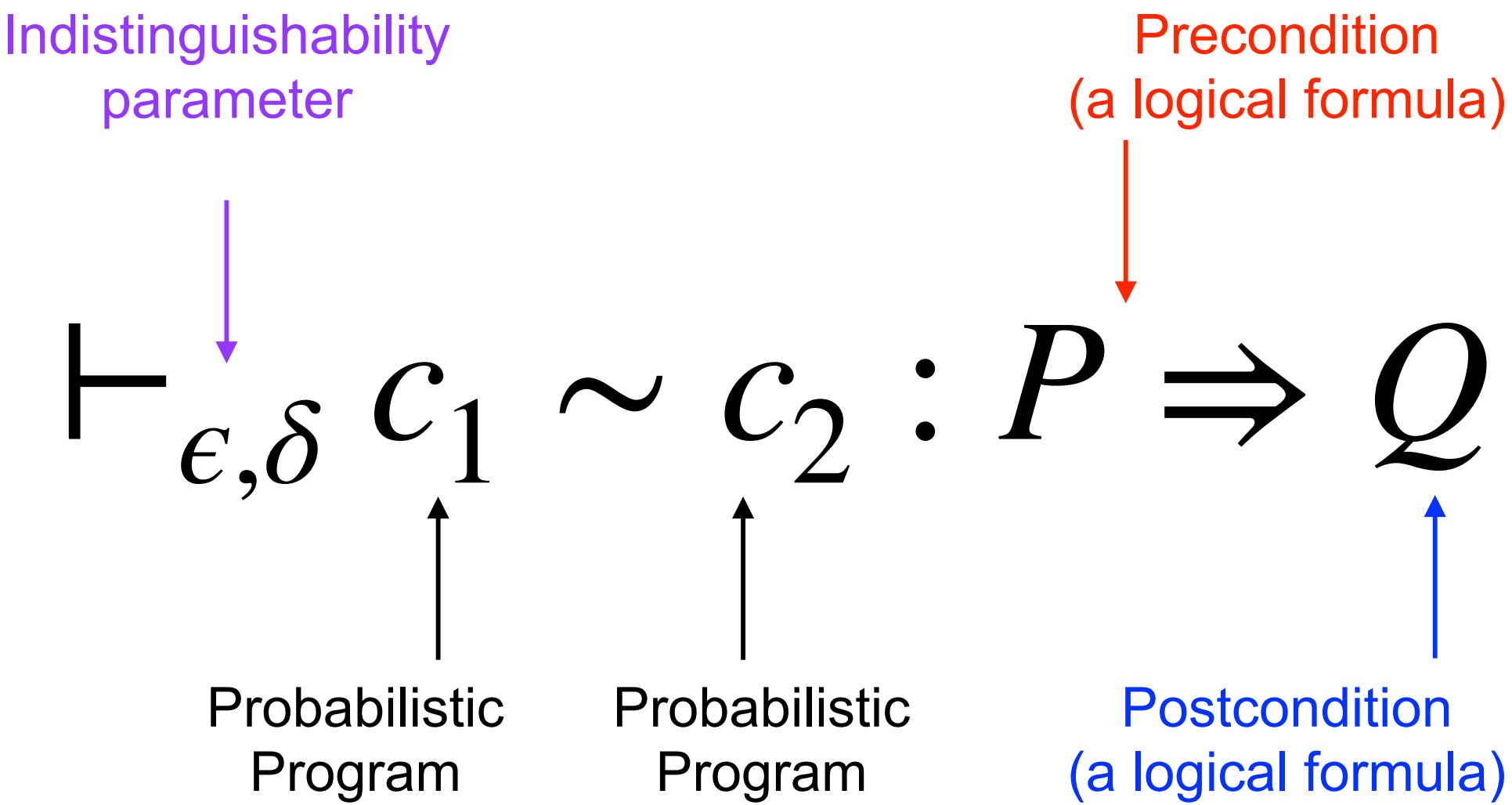
Definition

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: skip rule

$$\frac{}{\vdash_{0,0} \texttt{skip} \sim \texttt{skip} : P \Rightarrow P}$$

apRHL: More general Lap rule (still restricted)

$$\frac{}{\vdash_{k*\varepsilon,0} x_1 :=\$\ \mathrm{Lap}(1/\varepsilon, y_1) \sim x_2 :=\$\ \mathrm{Lap}(1/\varepsilon, y_2) : |y_1 - y_2| \leq k \Rightarrow =}$$

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

$$P /\backslash\ e\langle 1\rangle \leq 0 \Rightarrow \neg b1\langle 1\rangle$$

$$\vdash_{\varepsilon_k,\delta_k} c1 \sim c2 : P /\backslash b1\langle 1\rangle /\backslash b2\langle 2\rangle /\backslash k = e\langle 1\rangle /\backslash e\langle 1\rangle \leq n \Longrightarrow P /\backslash b1\langle 1\rangle = b2\langle 2\rangle /\backslash k < e\langle 1\rangle$$

$$\vdash_{\sum\varepsilon_k,\sum\delta_k} \text{while } b1 \text{ do } c1 \sim \text{while } b2 \text{ do } c2 : P /\backslash b1\langle 1\rangle = b2\langle 2\rangle /\backslash e\langle 1\rangle \leq n \Longrightarrow P /\backslash \neg b1\langle 1\rangle /\backslash \neg b2\langle 2\rangle$$

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

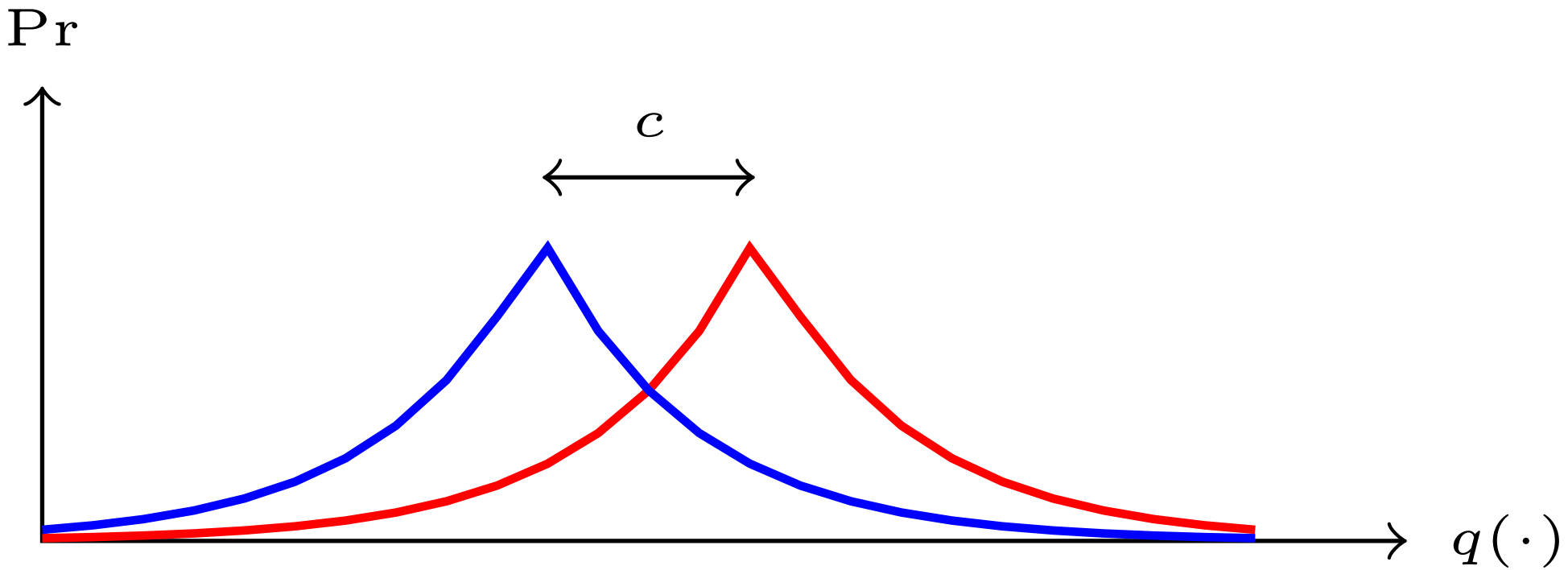
```
DummySum(d : {0,1} list) : real list
   i:=0;
   s:=0;
   r:=[];
   while (i<size d)
      z:=$ Lap(eps,d[i])
      s:= s + z
      r:= r ++ [s];
      i:= i+1;
   return r
```

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.

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

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.

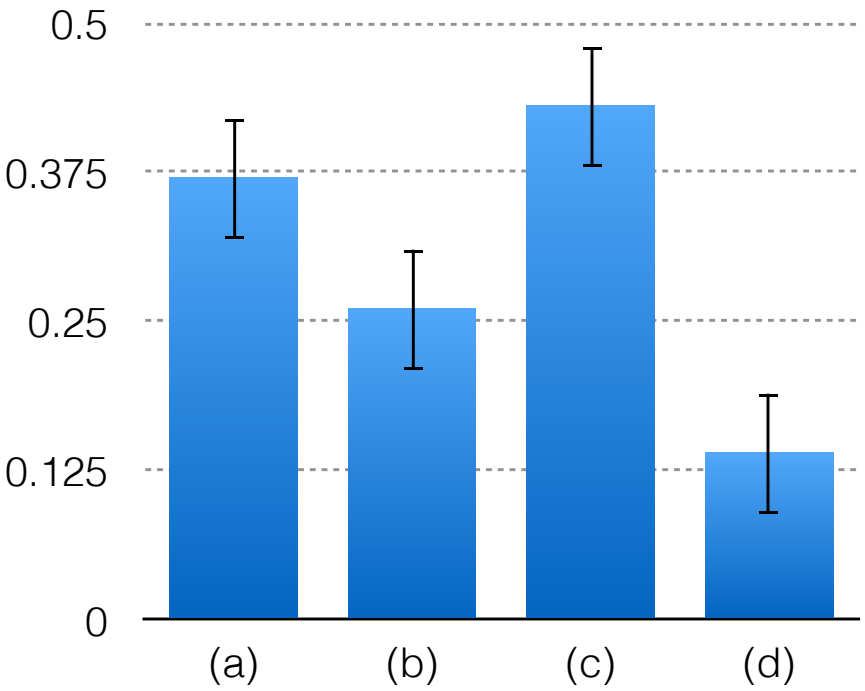
Report Noisy Max

(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 l, 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 l, 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 $k\varepsilon$-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

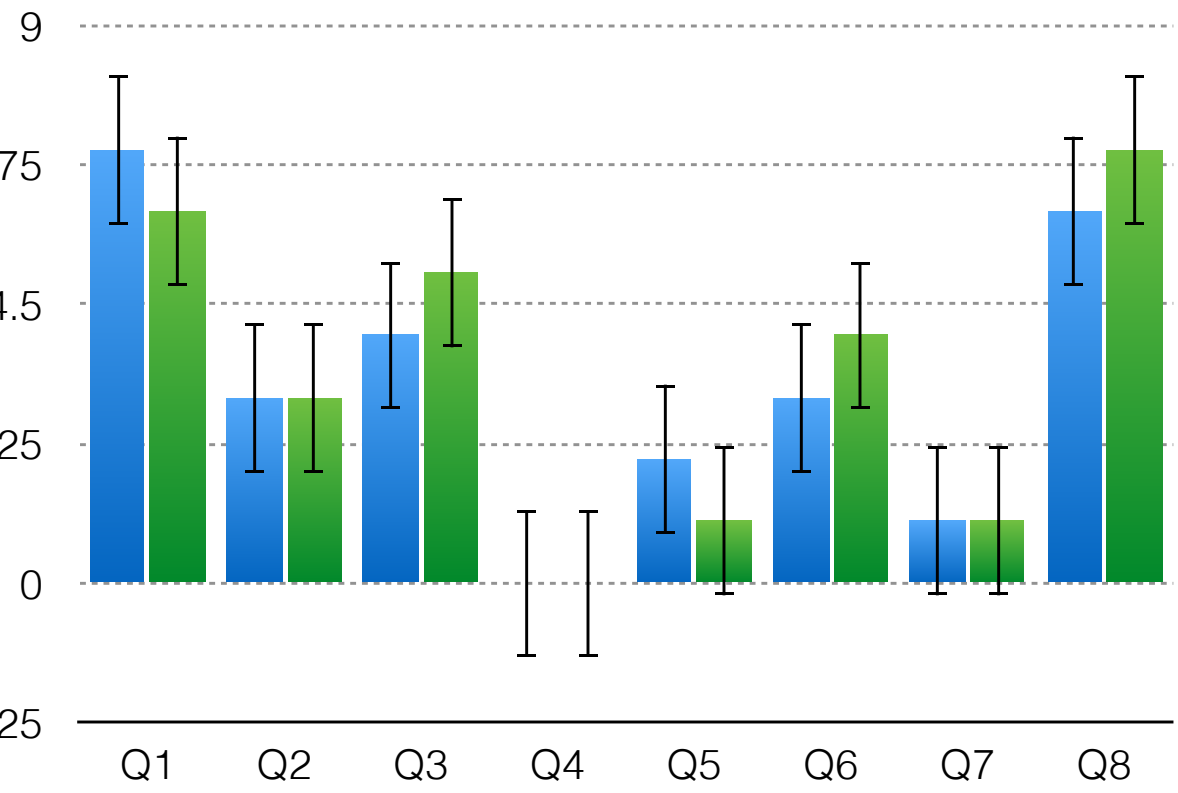
l sensitive queries

$q_1(D)$+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



I 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

....

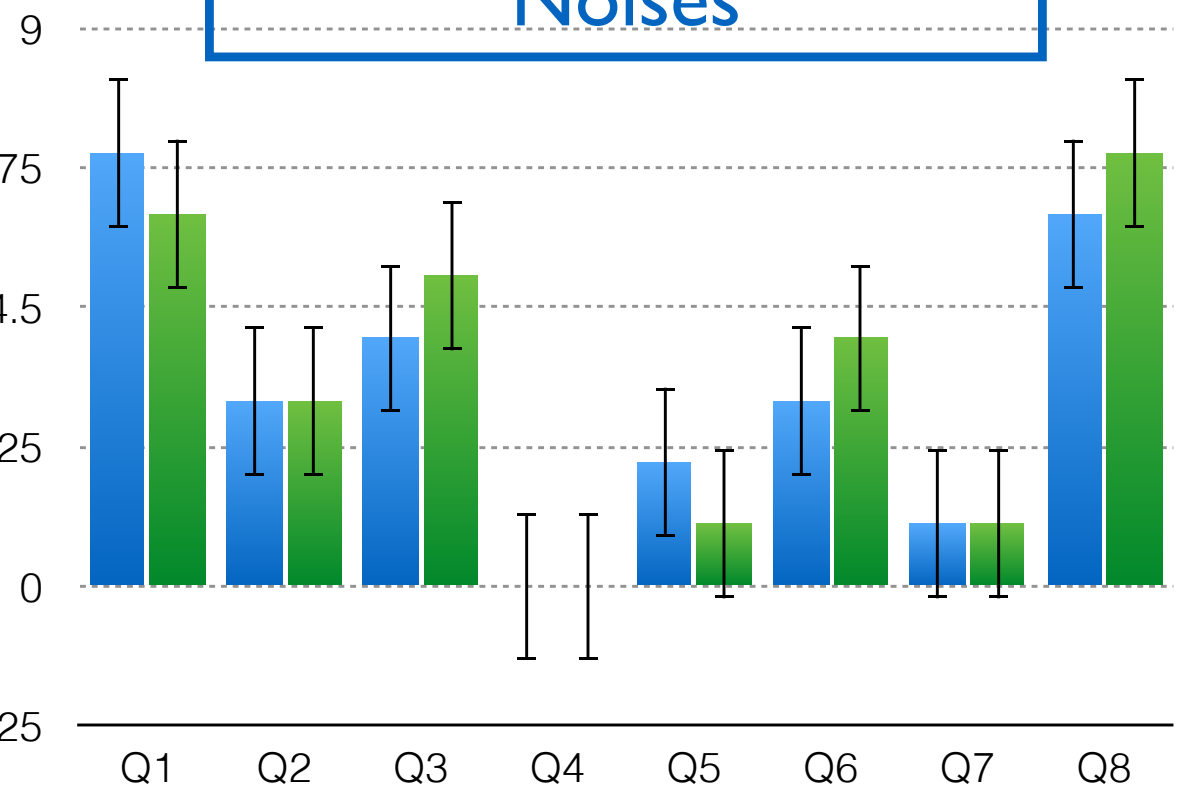
$q_k(D)$+noise $q_k(D')$+noise

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

l 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

Algorithm 8 Pseudo-code for Report Noisy Max

1: **function** RNM(D, $q_1, \ldots, q_m$, ϵ)
2: **for** $i \leftarrow 1, \ldots, m$ **do**
3: $c_i \leftarrow \mathsf{LapMech}(D, q_i, \epsilon)$
4: **end for**
5: **return** $\mathsf{argmax}_i c_i$
6: **end function**

Report Noisy Max

Theorem 1.12. The algorithm RNM is ϵ-differentially private.

Report Noisy Max

Theorem 1.12. The algorithm RNM is ϵ-differentially private.

Proof. Fix $D \sim D'$, and $c = q(\vec{D})$ and $c' = q(\vec{D}')$ be the results of the counting queries in the two cases. For simplicity we assume that c and c' have the following properties.

- i) $\forall j, c_j \geq c'_j$
- ii) $\forall j, 1 + c'_j \geq c_j$

We consider non-normalized queries

The general case is a little more involved but overall similar.

Let's now fix i, we want to bound the ratio of the probability that i is selected when running RNM on D and D'. Now let r_j be the noise drawn from Laplace for the round j and r_{-i} be the noise drawn from Laplace for all the rounds except the i-th one.

Report Noisy Max

Theorem 1.12. The algorithm RNM is ϵ-differentially private.

We first argue that $\Pr[i|D, r_{-i}] \leq e^{\varepsilon} \Pr[i|D', r_{-i}]$. Define

$$r^* = \min_{r_i} : c_i + r_i > c_j + r_j \ \forall j \neq i.$$

Note that, having fixed r_{-i}, i will be the output (the argmax noisy count) when the database is D if and only if $r_i \geq r^*$.
We have, for all $1 \leq j \neq i \leq m$:

$$\begin{aligned} c_i + r^* &> c_j + r_j \\ \Rightarrow (1 + c'_i) + r^* \geq c_i + r^* &> c_j + r_j \geq c'_j + r_j \\ \Rightarrow c'_i + (r^* + 1) &> c'_j + r_j. \end{aligned}$$

Report Noisy Max

Theorem 1.12. The algorithm RNM is ϵ-differentially private.

Thus, if $r_i \geq r^* + 1$, then the ith count will be the maximum when the database is D' and the noise vector is (r_i, r_{-i}). The probabilities below are over the choice of $r_i \sim \text{Lap}(1/\varepsilon)$.

$$\Pr[r_i \geq 1 + r^*] \geq e^{-\varepsilon} \Pr[r_i \geq r^*] = e^{-\varepsilon} \Pr[i|D, r_{-i}]$$

$$\Rightarrow \Pr[i|D', r_{-i}] \geq \Pr[r_i \geq 1 + r^*] \geq e^{-\varepsilon} \Pr[r_i \geq r^*] = e^{-\varepsilon} \Pr[i|D, r_{-i}],$$

which, after multiplying through by e^{ε}, yields what we wanted to show: $\Pr[i|D, r_{-i}] \leq e^{\varepsilon} \Pr[i|D', r_{-i}]$.

Report Noisy Max

Theorem 1.12. The algorithm RNM is ϵ-differentially private.

We now argue that $\Pr[i|D', r_{-i}] \leq e^{\varepsilon} \Pr[i|D, r_{-i}]$. Define

$$r^* = \min_{r_i} : c'_i + r_i > c'_j + r_j \ \forall j \neq i.$$

Note that, having fixed r_{-i}, i will be the output (argmax noisy count) when the database is D' if and only if $r_i \geq r^*$.
We have, for all $1 \leq j \neq i \leq m$:

$$\begin{aligned}
c'_i + r^* &> c'_j + r_j \\
\Rightarrow 1 + c'_i + r^* &> 1 + c'_j + r_j \\
\Rightarrow c'_i + (r^* + 1) &> (1 + c'_j) + r_j \\
\Rightarrow c_i + (r^* + 1) \geq c'_i + (r^* + 1) &> (1 + c'_j) + r_j \geq c_j + r_j.
\end{aligned}$$

Report Noisy Max

Theorem 1.12. The algorithm RNM is ϵ-differentially private.

Thus, if $r_i \geq r^* + 1$, then i will be the output (the argmax noisy count) on database D with randomness (r_i, r_{-i}). We therefore have, with probabilities taken over choice of r_i:

$$\Pr[i|D, r_{-i}] \geq \Pr[r_i \geq r^* + 1] \geq e^{-\varepsilon} \Pr[r_i \geq r^*] = e^{-\varepsilon} \Pr[i|D', r_{-i}],$$

which, after multiplying through by e^{ε}, yields what we wanted to show: $\Pr[i|D', r_{-i}] \leq e^{\varepsilon} \Pr[i|D, r_{-i}]$. □

Report One-sided Noisy Max

Instead of the classic Report Noisy Max, we consider a version where we add noise from a one-sided Laplace

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1;
  best = 0;
  while (i ≤ k){
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
            max = i ;
            best = cur;
   return max;
```

Report One-sided Noisy Max

Instead of the classic Report Noisy Max, we consider a version where we add noise from a one-sided Laplace

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1;
  best = 0;
  while (i ≤ k){
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
           max = i ;
           best = cur;
   return max;
```

Composition doesn't apply, since adding one-sided Laplace is not differentially private

Report One-sided Noisy Max

$\vdash_{-}(\varepsilon,0)$

$[b_1 \sim_1 b_2, \forall i.\forall d_1 \sim_1 d_2.\ |q_i(d_1)-q_i(d_2)| \leq 1, \ldots]$

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
  while (i ≤ k){
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
           max = i ;
           best = cur;
      i=i+1;
   return max;
```

$[max_1 = max_2]$

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?

Pointwise rule - simplified

If for every $s \in O$

$\vdash_{(\varepsilon, \delta_s)}$ Pre: $b_1 \sim_1 b_2$

program

Post: [out1=s => out2=s]

and

$\sum_{s \in O} \delta_s \leq \delta$

then

$\vdash_{(\varepsilon, \delta)}$ Pre: formula

Program

Post: [out_1=out_2]

Pointwise rule - simplified

If for every $s \in O$

$\vdash_{(\varepsilon, \delta_s)}$ Pre: $b_1 \sim_1 b_2$

program

Post: [out1=s => out2=s]

and

$\sum_{s \in O} \delta_s \leq \delta$

then

$\vdash_{(\varepsilon, \delta)}$ Pre: formula

Program

Post: [out_1=out_2]

Pointwise DP in Aprhl

$$\frac{\text{forall } r \in R \quad \vdash_{\varepsilon, \delta r} c_1 \sim c_2 : P \Longrightarrow x\langle 1\rangle = r \Rightarrow x\langle 2\rangle = r \qquad \sum \delta r \leq \delta}{\vdash_{\varepsilon, \delta} c_1 \sim c_2 : P \Longrightarrow x\langle 1\rangle = x\langle 2\rangle}$$

Report One-sided Noisy Max

$\vdash_{-}(\varepsilon,0)$

$[b_1 \sim_1 b_2, \forall i.\forall d_1 \sim_1 d_2.\ |q_i(d_1)-q_i(d_2)| \leq 1, \ldots]$

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
  while (i ≤ k){
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
           max = i ;
           best = cur;
      i=i+1;
   return max;
```

$[max_1 = s \Rightarrow max_2 = s]$

By applying the pointwise rule we get a different post

Report One-sided Noisy Max

$\vdash_{-}(\varepsilon,0)$

$[b_1 \sim_1 b_2, \forall i.\forall d_1 \sim_1 d_2.\ |q_i(d_1)-q_i(d_2)| \leq 1, \ldots]$

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
  while (i ≤ k){
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
           max = i ;
           best = cur;
      i=i+1;
   return max;
```

$[max_1=s \Rightarrow max_2=s]$

By applying the pointwise rule we get a different post

Notice that we focus on a single general s.

Report One-sided Noisy Max

$\vdash_{-}(\varepsilon,0)$

ROSNM $(q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat

$[b_1 \sim_1 b_2, \forall i.\forall d_1 \sim_1 d_2.\ |q_i(d_1)-q_i(d_2)| \leq 1, \ldots]$

```
i = 1; best = 0;
while (i ≤ k){
    cur = qi(b) + Lap+(ε/2)
    if (cur > best \/ i=1)
        max = i ;
        best = cur;
    i=i+1;
return max;
```

$[max_1=s \Rightarrow max_2=s]$

We can apply standard RHL

Report One-sided Noisy Max

```
|-(ε,0)
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
[b1 ~1 b2,…]
  while (i ≤ k){
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
           max = i ;
           best = cur;
      i=i+1;
   return max;
[max1=s => max2=s]
```

We can apply standard RHL

Report One-sided Noisy Max

$\vdash_{-}(\varepsilon,0)$

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
 while (i ≤ k){
```

$[b_1 \sim_1 b_2, \ldots]$

```
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
            max = i ;
            best = cur;
      i=i+1;
   return max;
```

$[max_1 = s \Rightarrow max_2 = s]$

We can now proceed by cases

Report One-sided Noisy Max

$\vdash_{(\varepsilon,0)}$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat

i = 1; best = 0;
while (i ≤ k){

[$b_1 \sim_1 b_2$, $i_1<s \Rightarrow \ldots \wedge i_1>s \Rightarrow \ldots \wedge i_1=i_2$]

cur = q_i(b) + Lap+(ε/2)
if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;
return max;

[$max_1=s \Rightarrow max_2=s$]

And use different properties

Invariant

$$
\begin{aligned}
& i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1 \\
\wedge\ & i_1 \geq s \Rightarrow (max_1 = max_2 = s \wedge best_1 + 1 = best_2) \vee max_1 \neq s \\
\wedge\ & i_1 = i_2
\end{aligned}
$$

Invariant

This part describes the situation
before we encounter s.

$$i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1$$
$$\wedge\ i_1 \geq s \Rightarrow (max_1 = max_2 = s \wedge best_1 + 1 = best_2) \vee max_1 \neq s$$
$$\wedge\ i_1 = i_2$$

Invariant

$$
\begin{aligned}
& i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1 \\
\wedge\ & i_1 \geq s \Rightarrow (max_1 = max_2 = s \wedge best_1 + 1 = best_2) \vee max_1 \neq s \\
\wedge\ & i_1 = i_2
\end{aligned}
$$

This part describes the situation after we encounter s.

Invariant

$$
\begin{aligned}
& i_1 < s \Rightarrow max_1 < s \;\wedge\; max_2 < s \;\wedge\; |best_1 - best_2| \leq 1 \\
\wedge\; & i_1 \geq s \Rightarrow (max_1 = max_2 = s \;\wedge\; best_1 + 1 = best_2) \;\vee\; max_1 \neq s \\
\wedge\; & i_1 = i_2
\end{aligned}
$$

When we encounter s we switch
from one to the other

Report One-sided Noisy Max

$|-(\varepsilon,0)$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$[i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1]$

cur = q_i(b) + Lap+(ε)
if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;
return max;

$[max_1 = s \Rightarrow max_2 = s]$

Let us consider case by case

Report One-sided Noisy Max

$|-(\varepsilon,0)$

ROSNM $(q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$[i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1]$

cur = q_i(b) + Lap+(ε)
if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;
return max;

$[max_1 = s \Rightarrow max_2 = s]$

Which rule shall we apply?

Laplace+ rule I

$\vdash_{(\varepsilon,0)}$ Pre: true

output = input + Lap+(ε)

Post: [output1−output2=input1−input2]

Report One-sided Noisy Max

$\vdash_{-(\varepsilon,0)}$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat

```
  i = 1; best = 0;
 while (i ≤ k){
```

$\vdash_{-(0,0)}$

```
    cur = q_i(b) + Lap+(ε/2)
```

$[i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1 \wedge$
$cur1 - cur2 = q_i(b) - q_i(b)]$

```
    if (cur > best \/ i=1)
          max = i ;
          best = cur;
     i=i+1;
  return max;
```

$[max_1 = s \Rightarrow max_2 = s]$

Let's apply the rule

Report One-sided Noisy Max

$\vdash(\varepsilon,0)$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$\vdash(0,0)$

cur = q_i(b) + Lap+(ε/2)

$[i_1 < s \Rightarrow max_1 < s \wedge max_2 < s \wedge |best_1 - best_2| \leq 1 \wedge |cur1 - cur2| \leq 1]$

if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;
return max;

$[max_1 = s \Rightarrow max_2 = s]$ and paid ε

And rewrite…

Report One-sided Noisy Max

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
 while (i ≤ k){
|-(0,0)
     cur = qi(b) + Lap+(ε/2)
    if (cur > best \/ i=1)
            max = i ;
            best = cur;
[i1 < s => max1 < s /\ max2 < s /\ |best1-best2|≤1/\
   |cur1-cur2|≤1]
      i=i+1;
   return max;
[max1=s => max2=s] and paid ε
```

Proceeding…

Report One-sided Noisy Max

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
 while (i ≤ k){
|-(0,0)
     cur = qi(b) + Lap+(ε/2)
    if (cur > best \/ i=1)
            max = i ;
            best = cur;
[i1 < s => max1 < s /\ max2 < s /\ |best1-best2|≤1/\
   |cur1-cur2|≤1] paid 0
      i=i+1;
   return max;
[max1=s => max2=s] and paid ε
```

This preserves the invariant

Report One-sided Noisy Max

```
ROSNM (q1,…,qk : list data → R,
      b : list data, ε: R) : nat
  i = 1; best = 0;
 while (i ≤ k){
[i1 ≥ s =>(max1= max2=s /\ best1+1=best2) \/ max1 ≠ s]
      cur = qi(b) + Lap+(ε/2)
      if (cur > best \/ i=1)
            max = i ;
            best = cur;
      i=i+1;
   return max;
[max1=s => max2=s] and paid ε
```

Let us consider now the second case

Report One-sided Noisy Max

```
ROSNM (q1,…,qk : list data → R,
       b : list data, ε: R) : nat
  i = 1; best = 0;
 while (i ≤ k){
[i1 ≥ s =>(max1= max2=s /\ best1+1=best2) \/ max1 ≠ s]
       cur = qi(b) + Lap+(ε/2)
       if (cur > best \/ i=1)
            max = i ;
            best = cur;
       i=i+1;
   return max;
[max1=s => max2=s] and paid ε
```

What rule shall we apply now?

Laplace+ rule 2

$\vdash_{(k'\varepsilon,0)}$ Pre: $0 \le k + input_1 - input_2 \le k'$

output = input + Lap+(ε)

Post: [output1+k=output2]

apRHL
Generalized Laplace

$x_1 :=\$\ \mathrm{Lap}(\varepsilon, e_1)$

$\vdash_{k_2 * \varepsilon, 0}\ \sim$

$x_2 :=\$\ \mathrm{Lap}(\varepsilon, e_2)$

$: \ |k_1 + e_1\langle 1\rangle - e_2\langle 2\rangle| \leq k_2$

$\Longrightarrow\ x1\langle 1\rangle + k_1 = x\langle 2\rangle$

Report One-sided Noisy Max

$\vdash(\varepsilon,0)$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$\vdash(\varepsilon,0)$

cur = q_i(b) + Lap+(ε/2)

$[(i_1 \geq s \Rightarrow (max_1 = max_2 = s \wedge best_1+1 = best_2) \vee max_1 \neq s) \wedge cur_1+1 = cur_2]$

if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;
return max;

$[max_1 = s \Rightarrow max_2 = s]$ and paid ε

Let's apply the rule

Report One-sided Noisy Max

$\vdash_{-}(\varepsilon,0)$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$\vdash_{-}(\varepsilon,0)$

cur = q_i(b) + Lap+(ε/2)

$[(i_1 \geq s \Rightarrow (max_1 = max_2 = s \wedge best_1+1 = best_2) \vee max_1 \neq s) \wedge cur_1+1 = cur_2]$

if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;
return max;

$[max_1 = s \Rightarrow max_2 = s]$ and paid ε

Now we see that either we don't enter the if in the first case, or if we do, we are guaranteed to enter also in the second case

Report One-sided Noisy Max

$\vdash(\varepsilon,0)$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$\vdash(\varepsilon,0)$

cur = q_i(b) + Lap+(ε/2)
if (cur > best \/ i=1)
max = i ;
best = cur;

$[(i_1 \geq s \Rightarrow (max_1 = max_2 = s \wedge best_1+1 = best_2) \vee max_1 \neq s) \wedge cur_1+1 = cur_2]$

i=i+1;

Continuing…

return max;

$[max_1 = s \Rightarrow max_2 = s]$ and paid ε

Report One-sided Noisy Max

$\vdash(\varepsilon,0)$

ROSNM ($q_1,\ldots,q_k$: list data → R,
b : list data, ε: R) : nat
i = 1; best = 0;
while (i ≤ k){

$\vdash(\varepsilon,0)$

cur = q_i(b) + Lap+(ε/2)
if (cur > best \/ i=1)
max = i ;
best = cur;
i=i+1;

[invariant]

return max;

[max_1=s => max_2=s] and paid ε

With this we can conclude