CS 591: Formal Methods in Security and Privacy

Differential Privacy

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

Feedback

Please fill out the (late) mid-semester evaluation.

Recording

This is a reminder that we will record the class and we will post the link on Piazza.

This is also a reminder to myself to start recording!

From the previous classes

A more realistic example

```
StreamCipher(m : private msg[n]) : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

How can we prove this secure?

```
OneTimePad(m : private msg[n])
          : public msg[n]
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

~

```
StreamCipher(m : private msg[n])
              : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

How can we prove this secure?

```
OneTimePad(m : private msg[n])
          : public msg[n]
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

~

```
StreamCipher(m : private msg[n])
             : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

m

m

How can we prove this secure?

```
OneTimePad(m : private msg[n])
          : public msg[n]
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

~

```
StreamCipher(m : private msg[n])
             : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

$$m \downarrow m \oplus k$$

$$m$$

How can we prove this secure?

```
OneTimePad(m : private msg[n])
          : public msg[n]
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

~

```
StreamCipher(m : private msg[n])
            : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

$$m \longrightarrow m \oplus k$$

$$m \longrightarrow m \oplus pk$$

How can we prove this secure?

```
OneTimePad(m : private msg[n])
          : public msg[n]
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

~

```
StreamCipher(m : private msg[n])
              : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

$$m \longrightarrow m \oplus k \qquad\qquad m \longrightarrow m \oplus pk$$

$$\Delta(\, m \oplus k \;,\; m \oplus pk \,) \leq \delta$$

Approximate Probabilistic Relational Hoare Logic

Indistinguishability parameter

Precondition (a logical formula)

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

Probabilistic Program

Probabilistic Program

Postcondition (a logical formula)

Validity of approximate Probabilistic Hoare judgments

We say that the quadruple $\vdash_\delta \mathtt{c}_1 \sim \mathtt{c}_2 : P \Rightarrow Q$ is valid if and only if for every pair of memories $\mathtt{m}_1, \mathtt{m}_2$ such that $P(\mathtt{m}_1, \mathtt{m}_2)$ we have:

$\{\mathtt{c}_1\}_{\mathtt{m}1} = \mu_1$ and $\{\mathtt{c}_2\}_{\mathtt{m}2} = \mu_2$ implies $Q_\delta{}^*(\mu_1, \mu_2)$.

$\mathtt{R}$–δ–Coupling

Given two distributions $\mu_1 \in D(A)$, and $\mu_2 \in D(B)$, we have an $\mathtt{R}$-δ-coupling between them, for $\mathtt{R} \subseteq A x B$ and $0 \leq \delta \leq \mathtt{1}$, if there are two joint distributions $\mu_{L,}\mu_R \in D(AxB)$ such that:

1) $\pi_1(\mu_L)=\mu_1$ and $\pi_2(\mu_R)=\mu_2$,
2) the support of μ_L and μ_R is contained in $\mathtt{R}$. That is, if $\mu_{\mathtt{L}}\mathtt{(a,b)>0,then\ (a,b)}\in\mathtt{R}$, $\mathtt{and}$ if $\mu_{\mathtt{R}}\mathtt{(a,b)>0,then\ (a,b)}\in\mathtt{R}$.
3) $\Delta(\mu_{L,}\mu_R) \leq \delta$

Probabilistic Relational Hoare Logic

Skip

$$\frac{}{\vdash_0 \texttt{skip} \sim \texttt{skip} : P \Rightarrow P}$$

Probabilistic Relational Hoare Logic

Composition

$$\frac{\vdash_{\delta_1} c_1 \sim c_2 : P \Rightarrow R \qquad \vdash_{\delta_2} c_1' \sim c_2' : R \Rightarrow S}{\vdash_{\delta_1+\delta_2} c_1;c_1' \sim c_2;c_2' : P \Rightarrow S}$$

Probabilistic Relational Hoare Logic

A specific rule for PRG

$\vdash_{2\text{^-}n}$ `x`$_1$ `:=$ Uniform({0,1}`^{n}`) ~`

`x`$_2$ `:=$ PRG(Uniform({0,1}`^{k}`))`

`: True` $\Rightarrow$ `x`$_1$`<1>=x`$_2$`<2>`

How can we prove this secure?

```
OneTimePad(m : private msg[n])
          : public msg[n]
   key :=$ Uniform({0,1}^n);
   cipher := msg xor key;
   return cipher
```

~

```
StreamCipher(m : private msg[n])
             : public msg[n]
   pkey :=$ PRG(Uniform({0,1}^k));
   cipher := msg xor pkey;
   return cipher
```

We can apply the PRG rule, the composition rule, and the assignment rule and prove:

$$\vdash_{2^{-n}} \texttt{OneTimePad} \sim \texttt{StreamCipher} : m\langle 1\rangle = m\langle 2\rangle \Rightarrow c\langle 1\rangle = c\langle 2\rangle$$

Differential Privacy

Data

Releasing the mean of Some Data

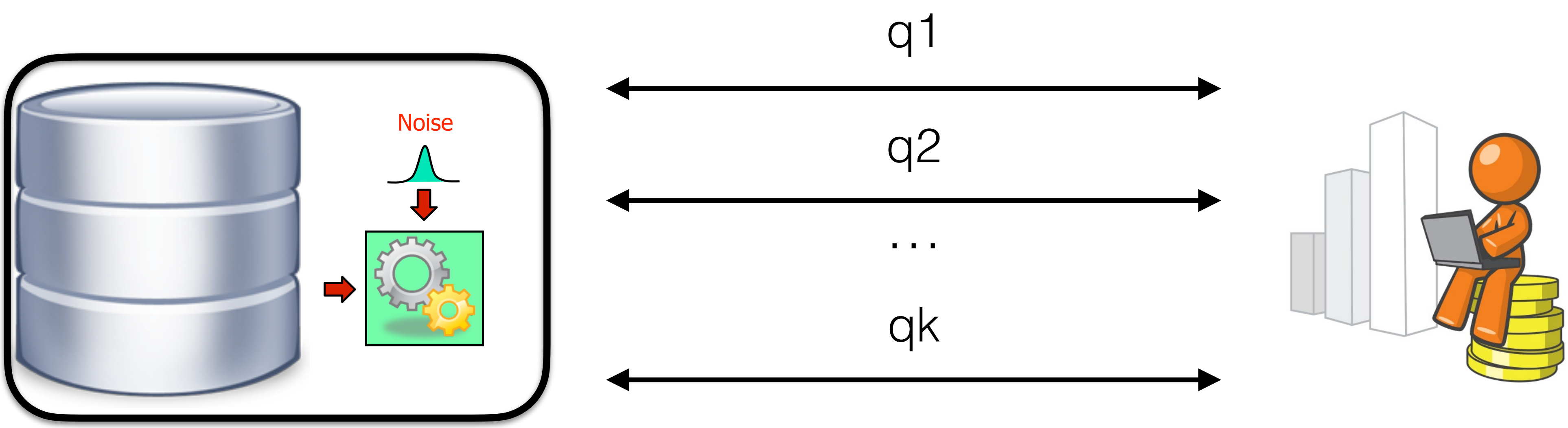
```
Mean(d : private data) : public real
   i:=0;
   s:=0;
   while (i<size(d))
      s:=s + d[i]
      i:=i+1;
   return (s/i)
```

Privacy-preserving data analysis?

We want to release some information to a data analyst and protect the privacy of the individuals contributing their data.

Privacy-preserving data analysis?

We want to release some information to a data analyst and protect the privacy of the individuals contributing their data.

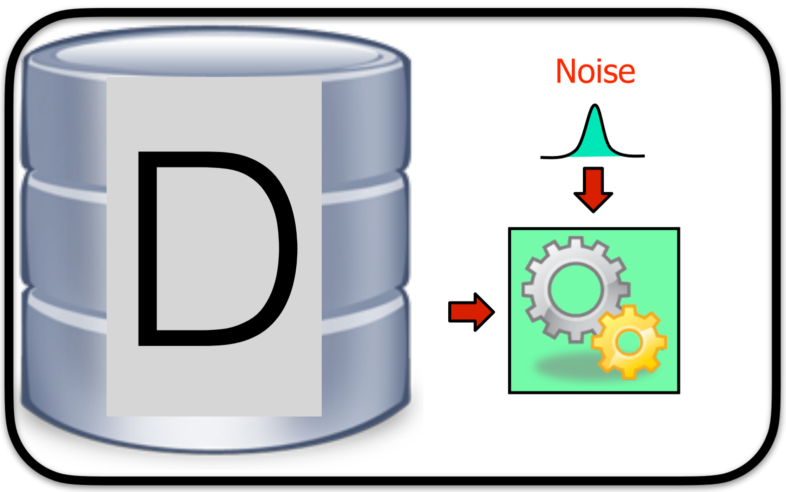


Data analyst

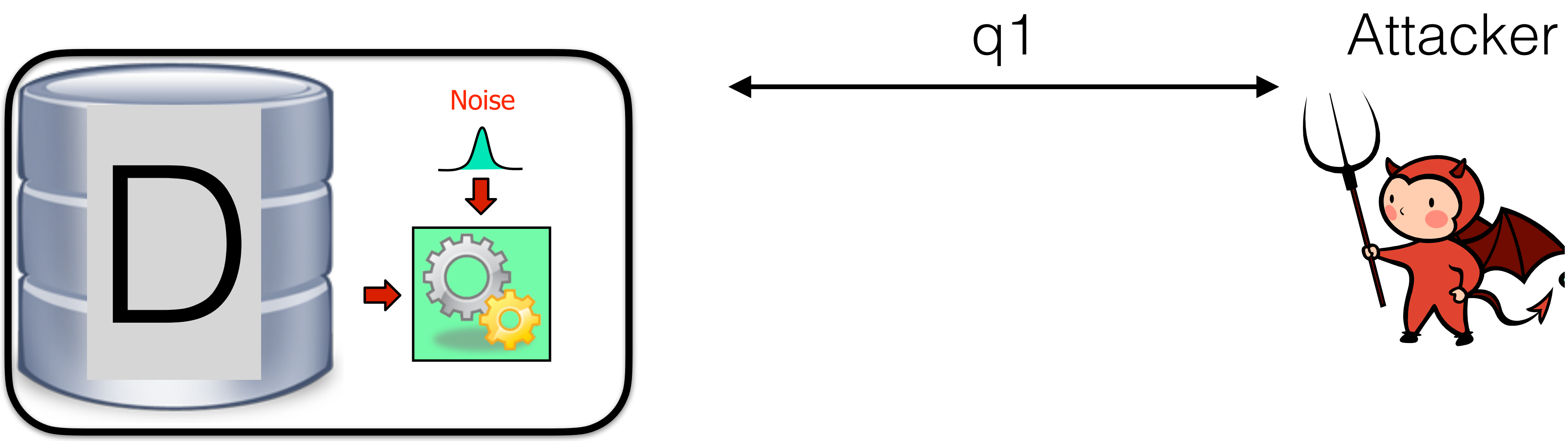
Fundamental Law of Information Reconstruction

The release of too many overly accurate statistics permits reconstruction attacks.

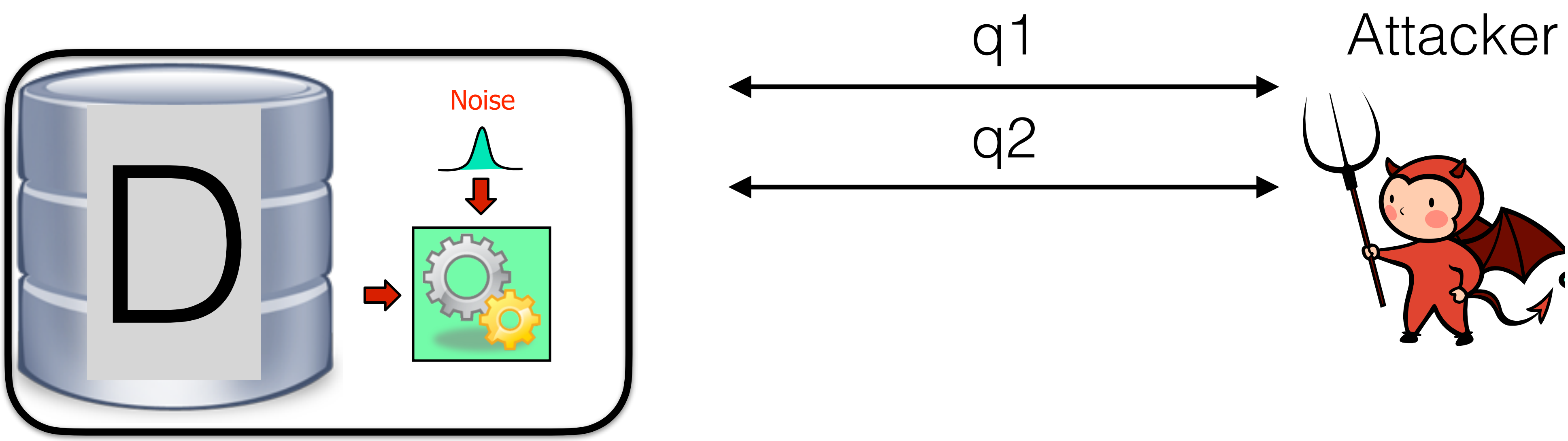
Reconstruction attack



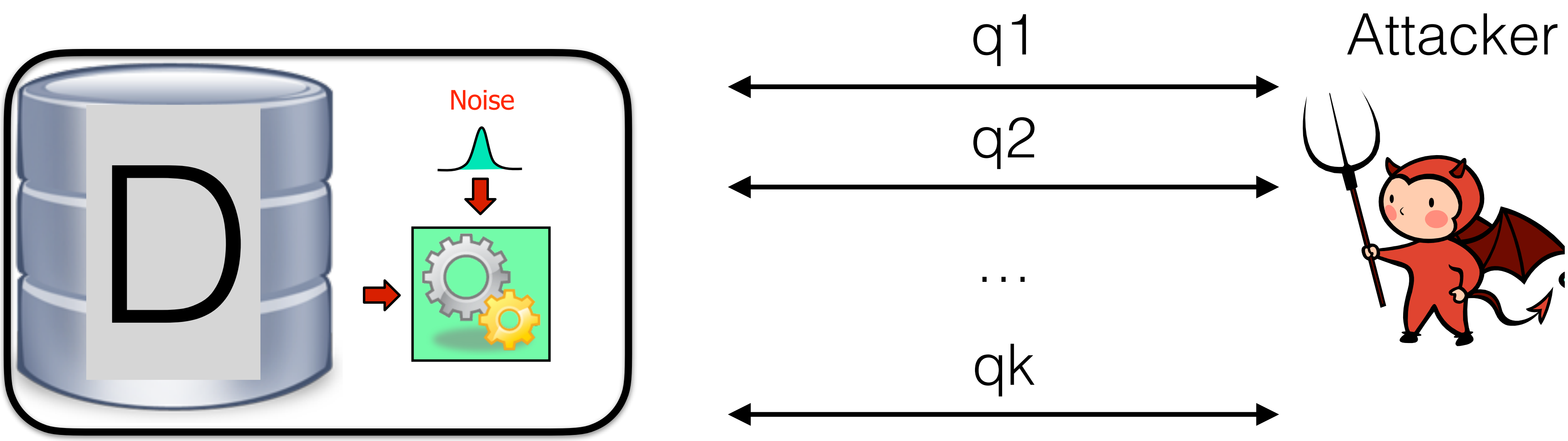
Reconstruction attack



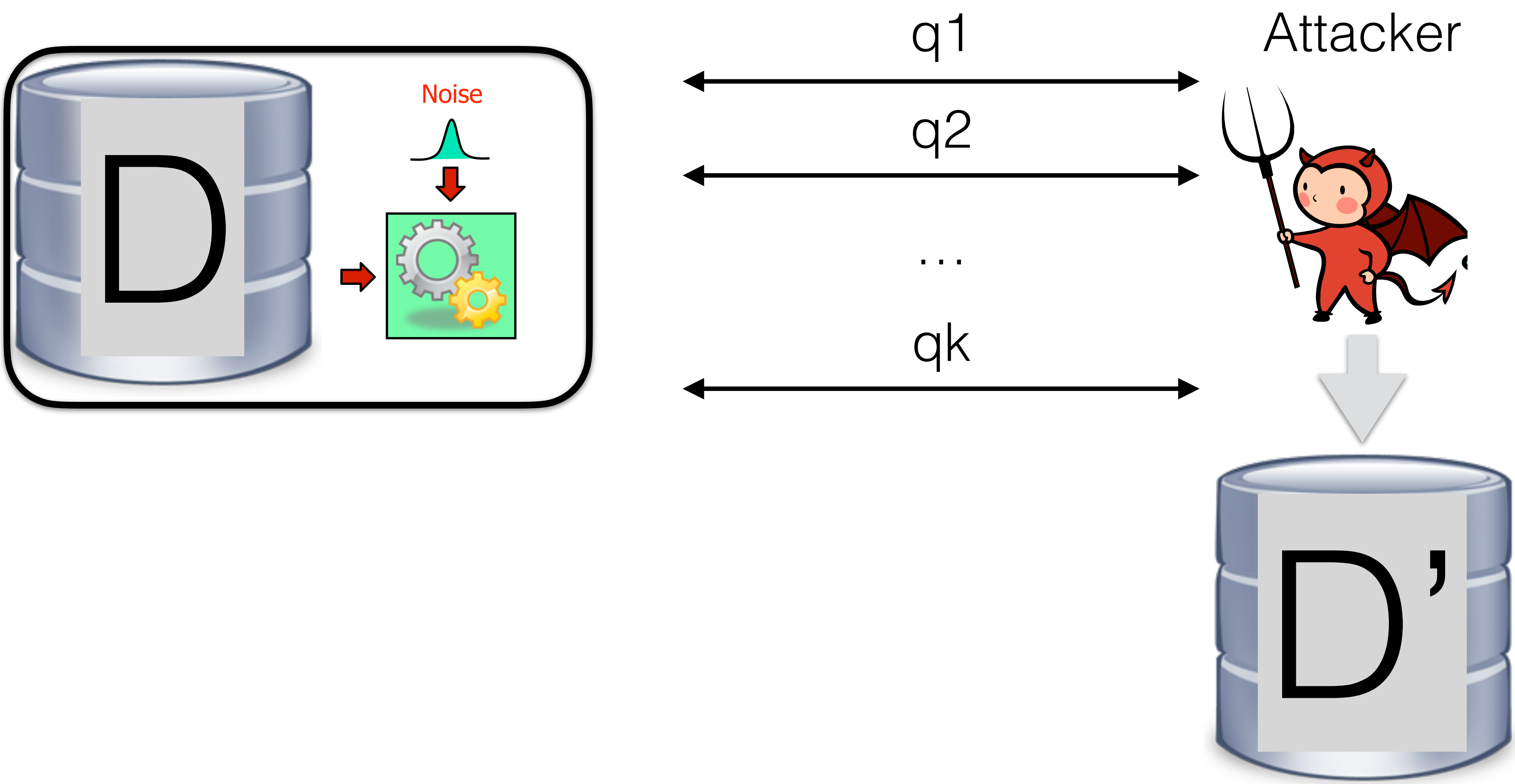
Reconstruction attack



Reconstruction attack



Reconstruction attack



Reconstruction attack

We say that the attacker wins if

$$d(D, D') \sim 0$$

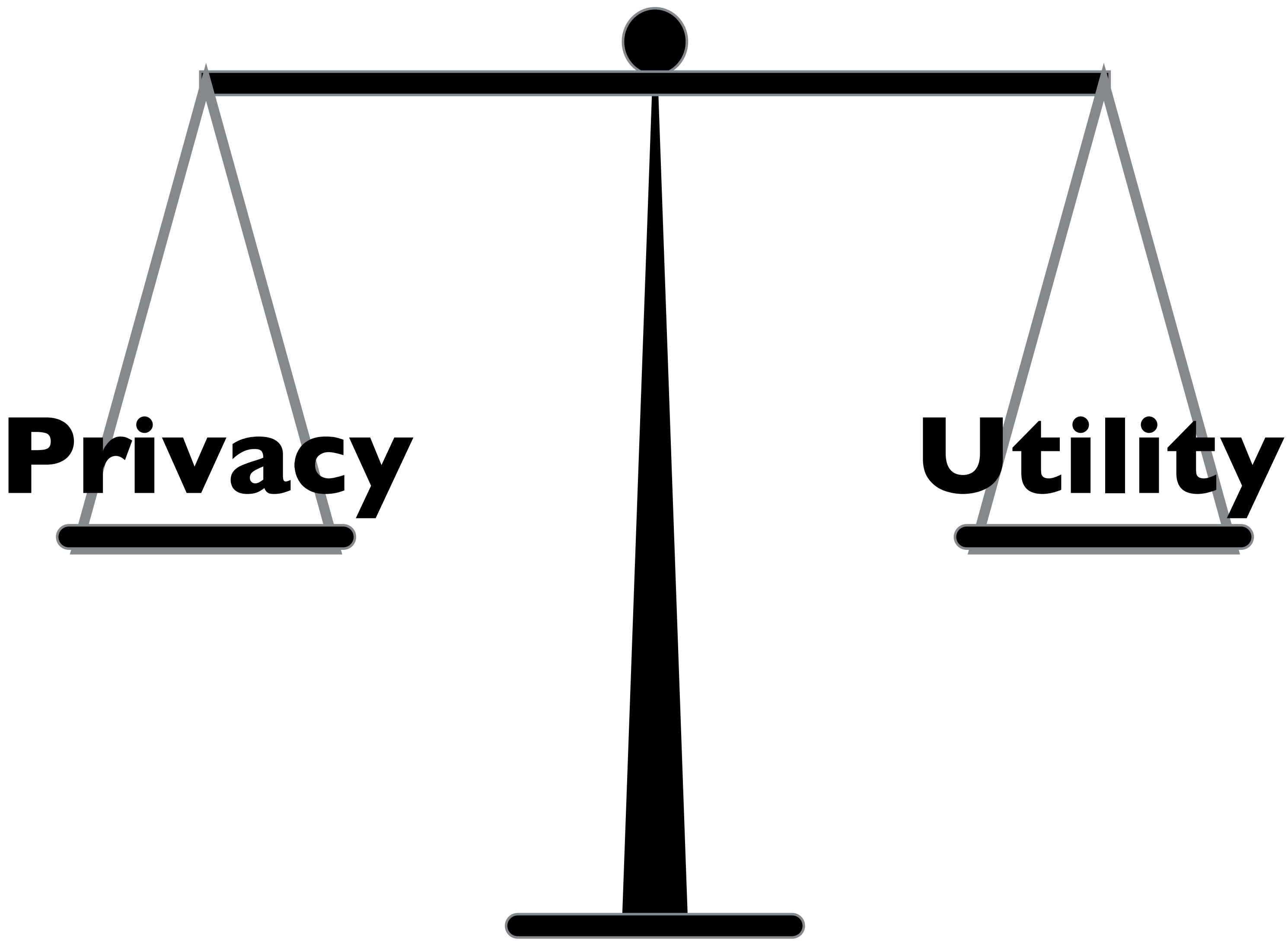
Reconstruction attack

We say that the attacker wins if

$$d(D, D') \sim 0$$

In this class case we can use Hamming distance

Privacy vs Utility



Quantitative notions of Privacy

- The impossibility results discussed above suggest a quantitative notion of privacy,
- a notion where the privacy loss depends on the number of queries that are allowed,
- and on the accuracy with which we answer them.

Differential privacy:
understanding the mathematical and computational meaning of this trade-off.

[Dwork, McSherry, Nissim, Smith, TCC06]

Privacy-preserving data analysis?

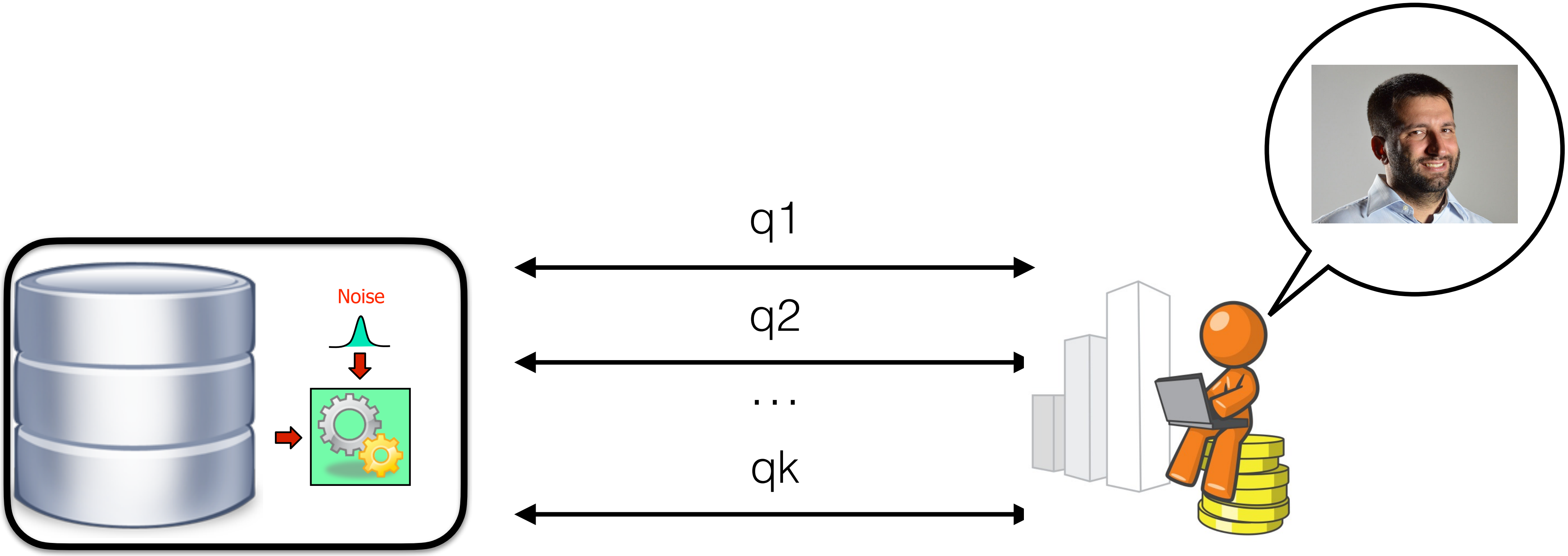
- The analyst knows no more about me after the analysis than what she knew before the analysis.

Privacy-preserving data analysis?

- The analyst knows no more about me after the analysis than what she knew before the analysis.

Privacy-preserving data analysis?

- The analyst knows no more about me after the analysis than what she knew before the analysis.



Privacy-preserving data analysis?

Prior Knowledge

~

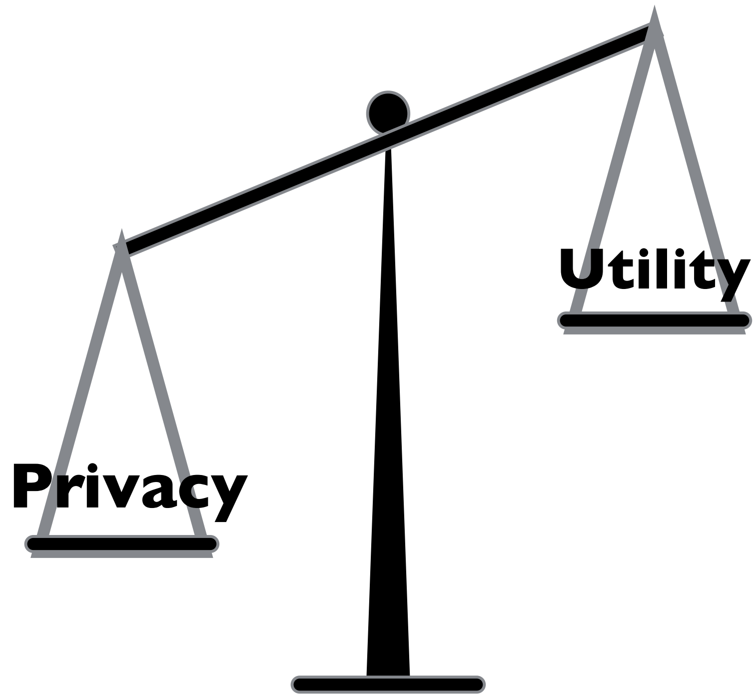
Posterior Knowledge

Privacy-preserving data analysis?

Privacy-preserving data analysis?

Question: What is the problem with this requirement?

Privacy-preserving data analysis?



If nothing can be learned about an individual, then nothing at all can be learned at all!

[DworkNaor10]

Privacy-preserving data analysis?

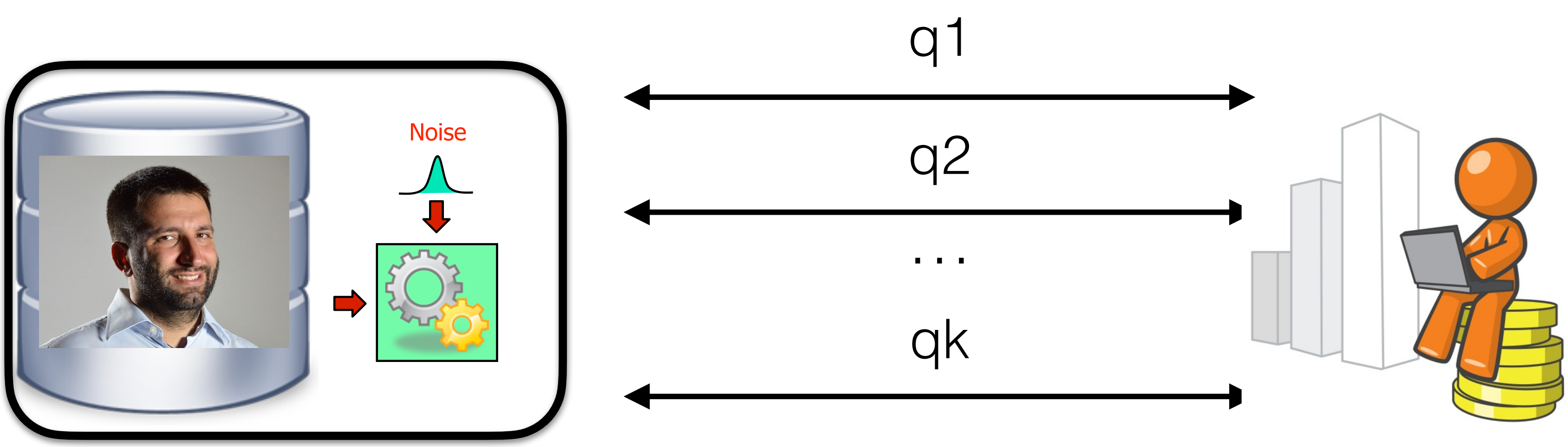
- The analyst learn almost the same about me after the analysis as what she would have learnt if I didn't contribute my data.

Privacy-preserving data analysis?

- The analyst learn almost the same about me after the analysis as what she would have learnt if I didn't contribute my data.

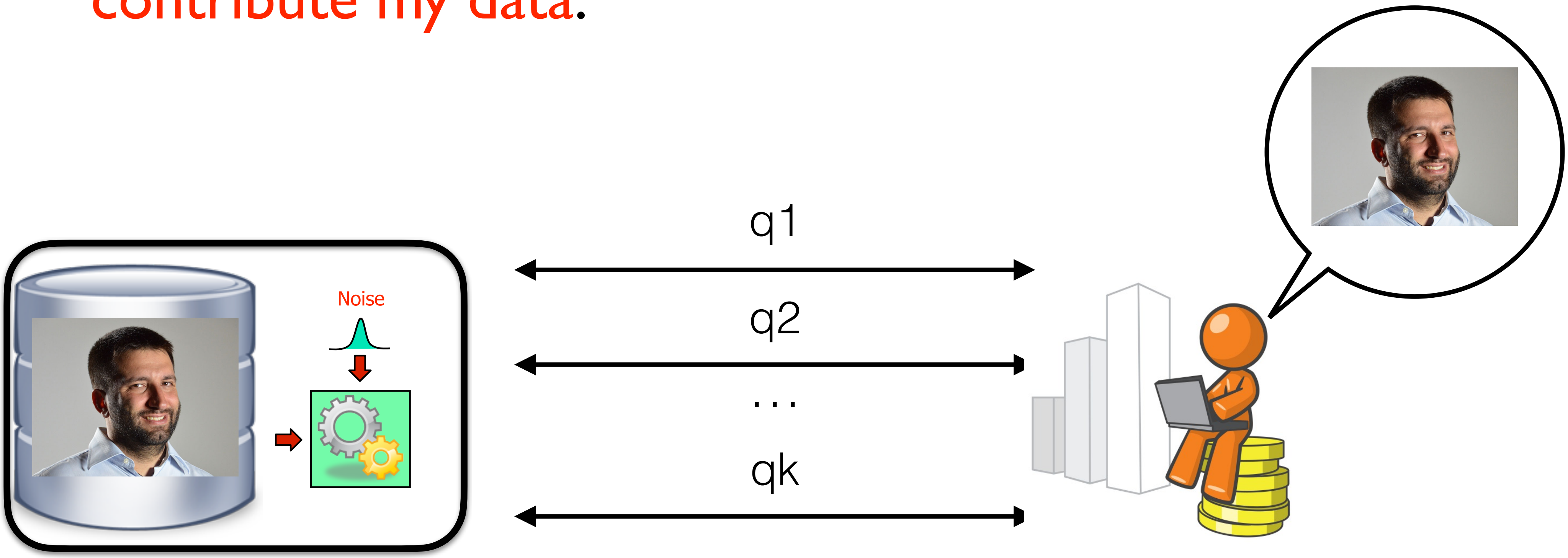
Privacy-preserving data analysis?

- The analyst learn almost the same about me after the analysis as what she would have learnt if I didn't contribute my data.



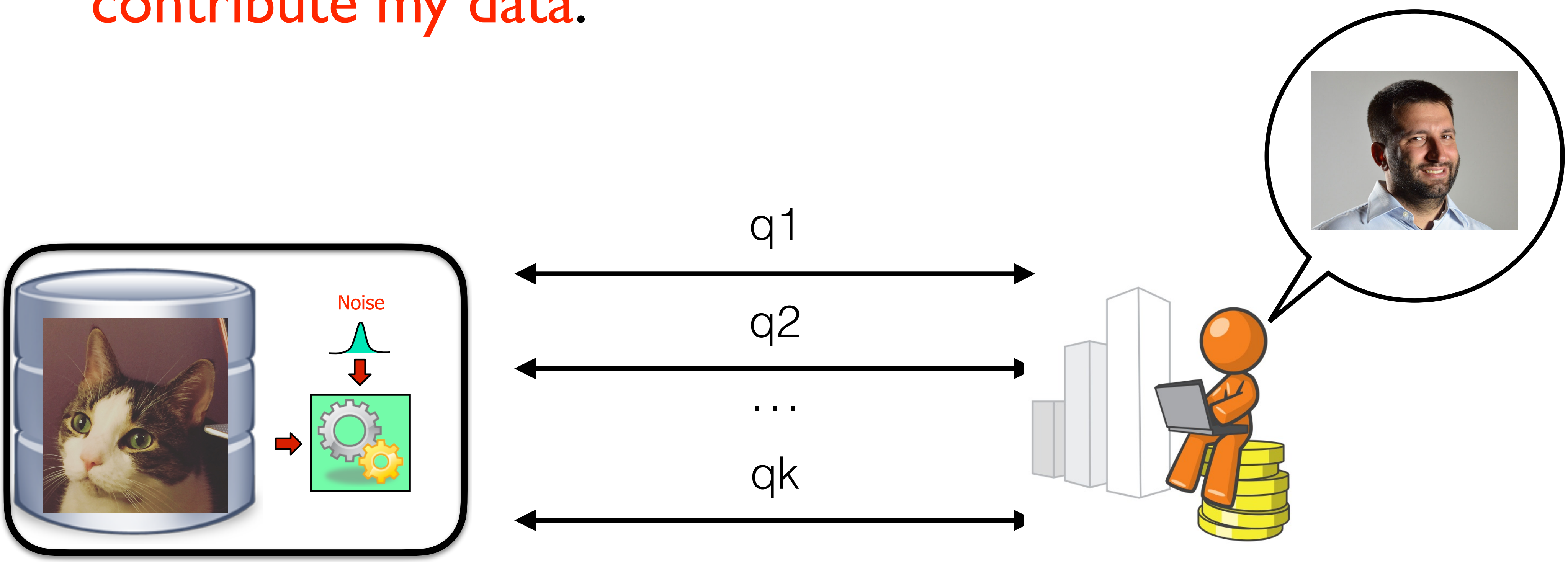
Privacy-preserving data analysis?

- The analyst learn almost the same about me after the analysis as what she would have learnt if I didn't contribute my data.



Privacy-preserving data analysis?

- The analyst learn almost the same about me after the analysis as what she would have learnt if I didn't contribute my data.



Adjacent databases

- We can formalize the concept of contributing my data or not in terms of a notion of distance between datasets.
- Given two datasets D, D'$\in$DB, their distance is defined as:

$$D\Delta D' = |\{k \leq n \mid D(k) \neq D'(k)\}|$$

- We will call two datasets adjacent when $D\Delta D' = 1$ and we will write $D \sim D'$.

Privacy Loss

In general we can think about the following quantity as the privacy loss incurred by observing r on the databases b and b'.

$$L_{b,b'}(r) = \log \frac{\Pr[Q(b)=r]}{\Pr[Q(b')=r]}$$

(ε,δ)-Differential Privacy

Definition

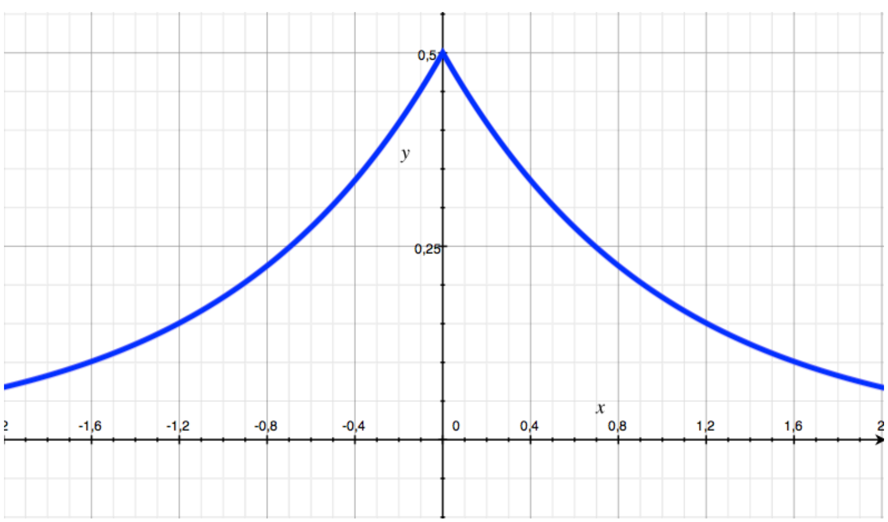
Given $\varepsilon,\delta \geq 0$, a probabilistic query $Q: X^n \rightarrow R$ is (ε,δ)-differentially private iff for all adjacent database b_1, b_2 and for every $S \subseteq R$:

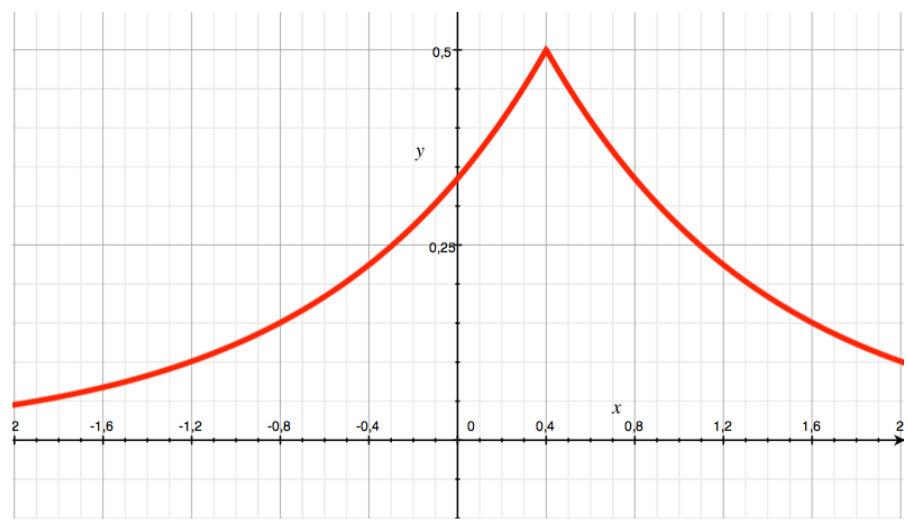
$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon)\Pr[Q(b_2) \in S] + \delta$$

Differential Privacy

Q : db => R   probabilistic

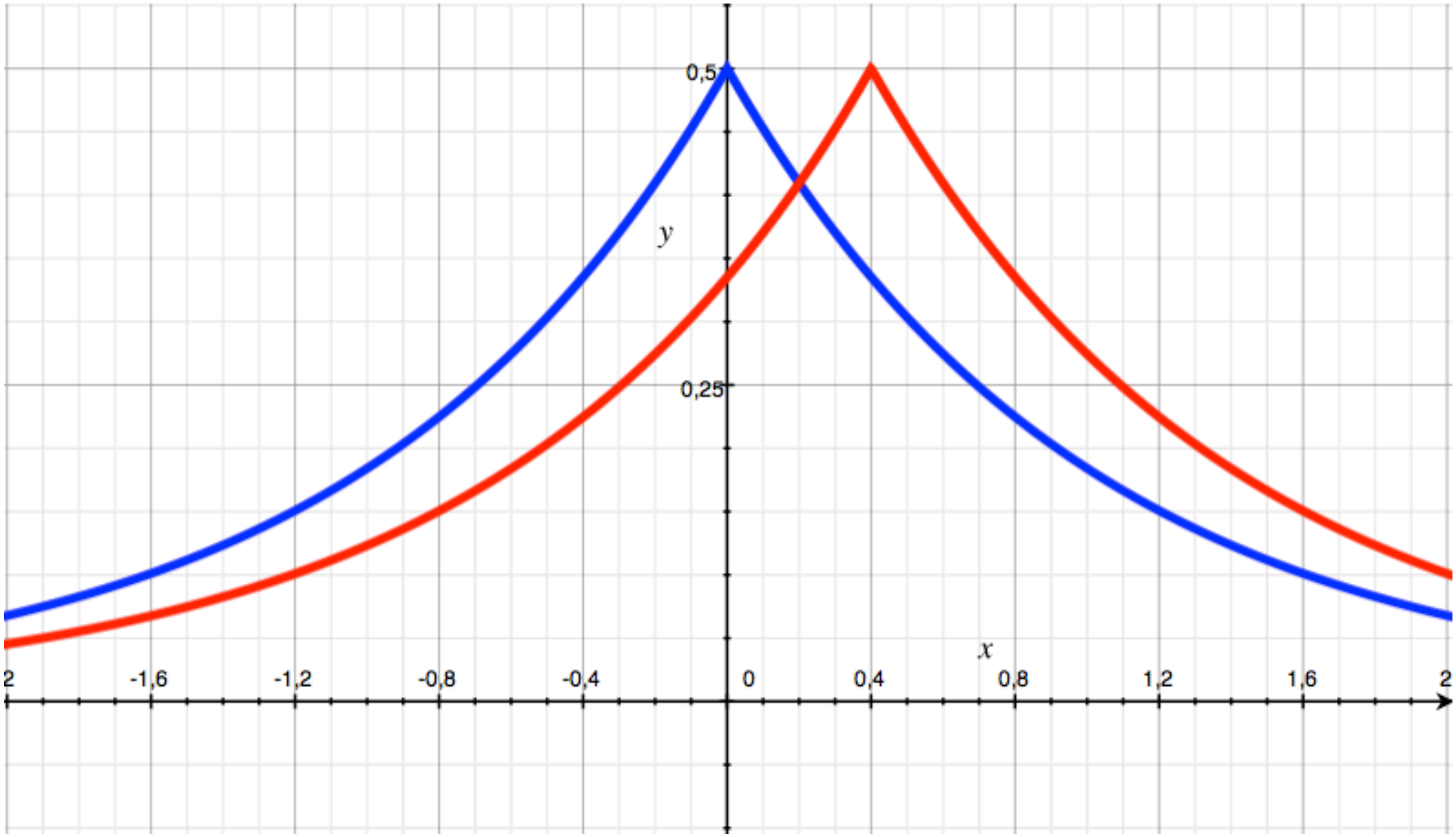
$Q(b \cup \{x\})$ $Q(b \cup \{y\})$





Differential Privacy

$d(Q(b \cup \{x\}), Q(b \cup \{y\})) \leq \varepsilon$ with probability $1-\delta$



(ε,δ)-Differential Privacy

$$\left| \log \frac{\Pr[Q(b_1)=r]}{\Pr[Q(b_2)=r]} \right| \leq \varepsilon \quad \text{with probability } 1-\delta$$