CS 511: Formal Methods in Security and Privacy

Differential Privacy

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

(ε,δ)-Differential Privacy

Definition

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

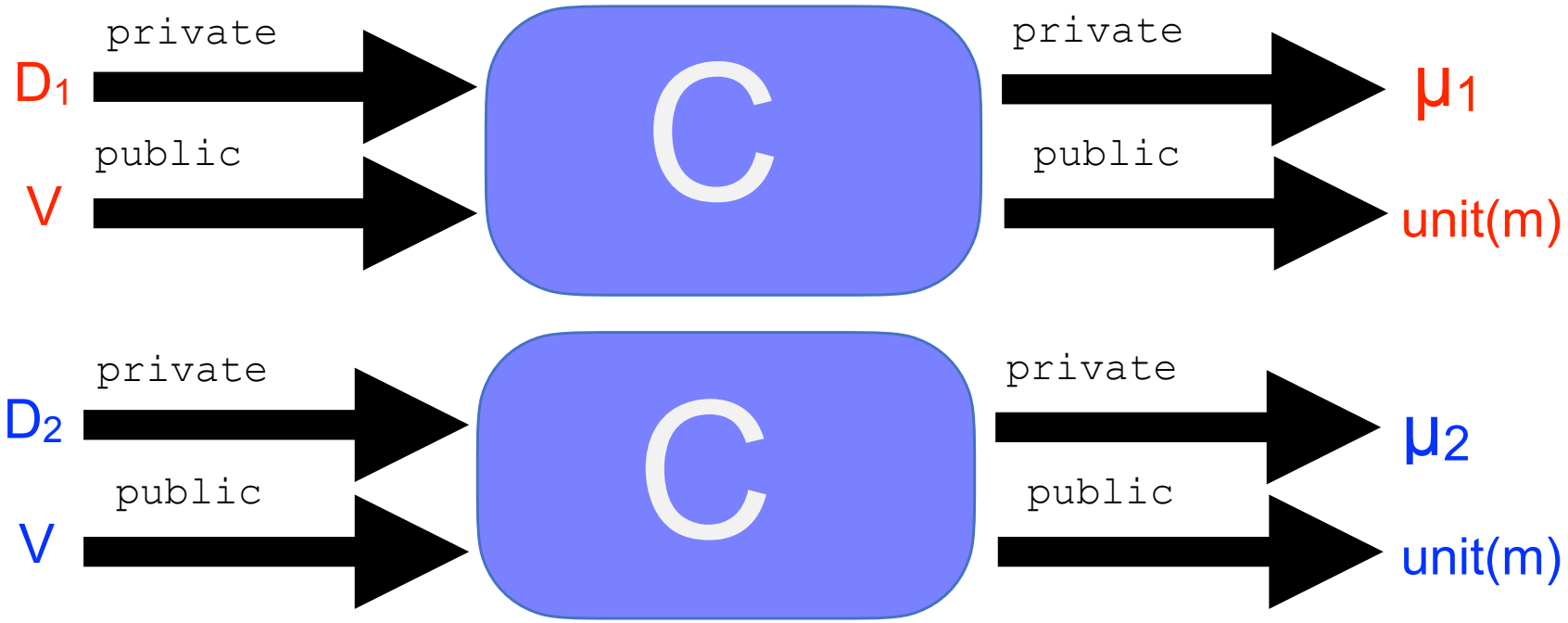
for all adjacent database b_1, b_2 and for every $S \subseteq R$:

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

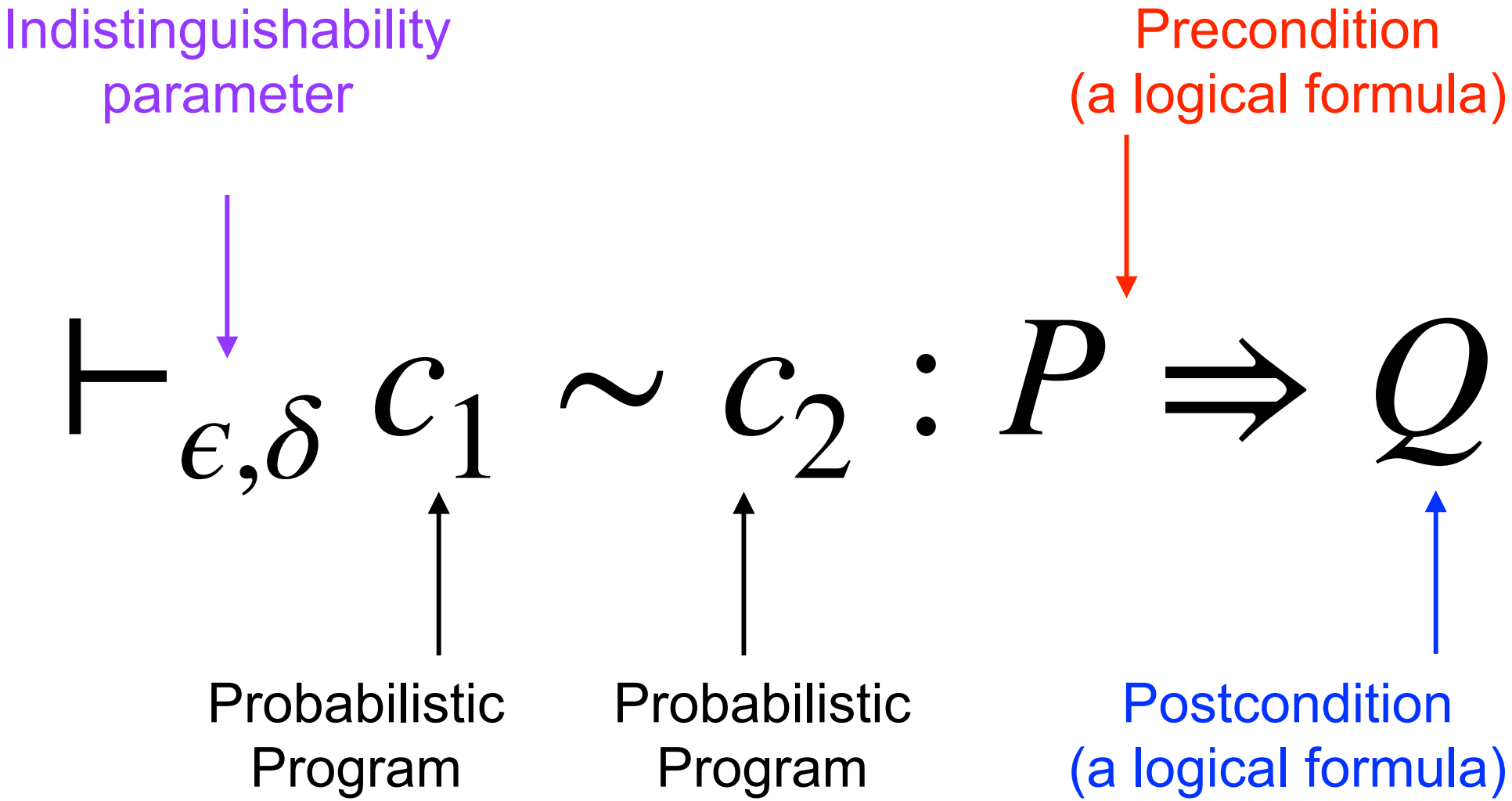
Differential Privacy as a Relational Property

c is differentially private if and only if for every $m_1 \sim m_2$ (extending the notion of adjacency to memories):

$\{c\}_{m1}=\mu_1$ and $\{c\}_{m2}=\mu_2$ implies $\Delta_\varepsilon(\mu_1,\mu_2) \leq \delta$



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)

Validity of apRHL judgments

We say that the 6-tuple $\vdash_{\varepsilon,\delta} c_1 \sim c_2 : P \Rightarrow Q$ is valid if and only if for every pair of memories m_1, m_2 such that $P(m_1, m_2)$ we have:

$\{c_1\}_{m1} = \mu_1$ and $\{c_2\}_{m2} = \mu_2$ implies

$Q_{\varepsilon,\delta}^*(\mu_1, \mu_2)$.

R–(ε,δ)–Coupling

Given two distributions $\mu_1 \in D(A)$, and $\mu_2 \in D(B)$, we have an R-(ε,δ)-coupling between them, for $R \subseteq A \times B$ and $0 \le \delta \le 1$, $\varepsilon \ge 0$, if there are two joint distributions $\mu_L, \mu_R \in D(A \times B)$ 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 R. That is, if $\mu_L(a,b) > 0$, then $(a,b) \in R$, and if $\mu_R(a,b) > 0$, then $(a,b) \in R$.
3) $\Delta_\varepsilon(\mu_L, \mu_R) \le \delta$

R-(ε,δ)-Coupling and Indistinguishability

Given two distributions $\mu_1 \in D(A)$, and $\mu_2 \in D(A)$, if we have a =-(ε,δ)-coupling between them, then they are (ε,δ)-indistinguishable.

So to show differential privacy it is enough to show:

$$\vdash_{\epsilon,\delta} c \sim c : \text{Adj} \Rightarrow \text{out}_1 = \text{out}_2$$

apRHL: skip rule

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

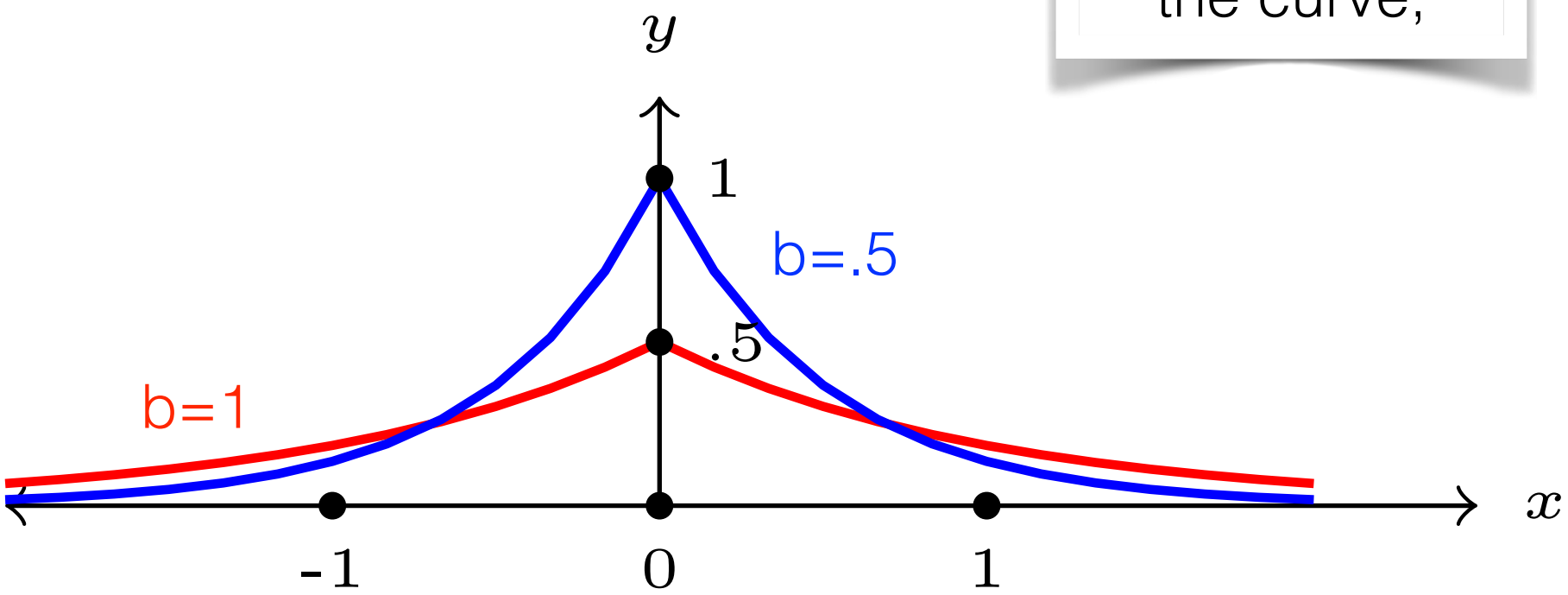
apRHL: Lap rule (simplified)

$$\vdash_{\varepsilon,0} \begin{array}{l} x_1 :=\$\ \mathrm{Lap}(1/\varepsilon, y_1) \\ \sim \\ x_2 :=\$\ \mathrm{Lap}(1/\varepsilon, y_2) \\ : \ |y_1 - y_2| \leq 1 \Rightarrow = \end{array}$$

Laplace Distribution

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

b regulates the skewness of the curve,



apRHL: More general Lap rule (still restricted)

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

Releasing privately the sum of Some Data

```
Mean(d : priv age [0,100]) : pub real
   i:=0;
   s:=0;
   while (i<size(d))
      s:=s + d[i]
      i:=i+1;
   z:=$ Lap(100/eps, s)
   return z
```

Releasing privately the average

```
Mean(d : priv age [0,100]) : pub real
   i:=0;
   s:=0;
   while (i<size(d))
      s:=s + d[i]
      i:=i+1;
   z:=$ Lap(100/eps, (s/i))
   return z
```

Releasing privately the average - v2

```
Mean(d : priv [n] age [0,100]) : pub real
   i:=0;
   s:=0;
   while (i<n)
      s:=s + d[i]
      i:=i+1;
   z:=$ Lap(100/(n*eps), (s/n))
   return z
```

Releasing privately two statistics

```
Mean(d : priv age [0,100]) : pub real*real
   i:=0;
   s:=0;
   while (i<size(d))
      s:=s + d[i]
      i:=i+1;
   z:=$ Lap(100/eps, s)
   y:=$ Lap(1/eps, i)
   return (z,y)
```

Composition

Let M_1:DB $\rightarrow R_1$ be a (ε_1,δ_1)-differentially private program and M_2:DB $\rightarrow R_2$ be a (ε_2,δ_1)-differentially private program. Then, their composition $M_{1,2}$:DB$\rightarrow R_1 x R_2$ defined as

$$M_{1,2}(D)=(M_1(D),M_2(D))$$

is $(\varepsilon_1+\varepsilon_2,\delta_1+\delta_2)$-differentially private.

Probabilistic Relational Hoare Logic
Composition

$$\frac{\vdash_{\varepsilon_1,\delta_1} c_1 \sim c_2 : P \Rightarrow R \qquad \vdash_{\varepsilon_2,\delta_2} c_1' \sim c_2' : R \Rightarrow S}{\vdash_{\varepsilon_1+\varepsilon_2,\delta_1+\delta_2} c_1; c_1' \sim c_2; c_2' : P \Rightarrow S}$$

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(1/eps,s)
      r:= r ++ [z];
      i:= i+1;
   return r
```