The opinions expressed in this course are mine and they do not reflect those of the National Science Foundation or the US. Census Bureau.
Differential privacy

**Definition**

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

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

This corresponds to a privacy loss of the form:

\[
\mathcal{L}_M^{D \rightarrow D'}(r) = \ln \left( \frac{\Pr[M(D) = r | E]}{\Pr[M(D') = r | E']} \right)
\]

The \((\varepsilon, \delta)\)-differential privacy requirement corresponds to requiring that for every \(r\) and every adjacent \(D, D'\) we have:

\[
\Pr \left[ \left| \mathcal{L}_M^{D \rightarrow D'}(r) \right| \leq \varepsilon \right] \geq 1 - \delta
\]

Not exactly!
Bounding the moments

A random variable can be described using its moments.

$$\mu_n = \mathbb{E}[X^n]$$

Here we consider central moments. For instance, the first central moment is the mean, the second is the variance, the third is the skewness, etc.

Can we bound the moments of the privacy loss?
Moment generating function

The probability distribution of a random variable $X$ can be described by its moment generating function:

$$m_X(\alpha) = \mathbb{E}[e^{\alpha X}]$$

This function can be used to compute, or give upper bounds on the moments of the random variable $X$.

$$m_X(\alpha) = 1 + \alpha \mu_1 + \frac{\alpha^2 \mu_2}{2!} + \ldots + \frac{\alpha^n \mu_n}{n!} + \ldots$$
Can we do better than the global sensitivity?
Global Sensitivity

Definition 1.8 (Global sensitivity). The **global sensitivity** of a function $q : \mathcal{X}^n \rightarrow \mathbb{R}$ is:

$$\Delta q = \max \left\{ |q(D) - q(D')| \mid D \sim_1 D' \in \mathcal{X}^n \right\}$$
1.7. The Laplace mechanism

is described in the following algorithm where $q : \mathbb{R}^{|X|} \rightarrow \mathbb{R}$ and where the notation $Y \overset{\$}{\leftarrow} \text{Lap}(\frac{\Delta q}{\epsilon})(0)$ denotes the fact that $Y$ is sampled from the distribution $\text{Lap}(\frac{\Delta q}{\epsilon})$.

Algorithm 2  Pseudo-code for the Laplace Mechanism

1: function $\text{LAPMECH}(D, q, \epsilon)$
2: \hspace{1cm} $Y \overset{\$}{\leftarrow} \text{Lap}(\frac{\Delta q}{\epsilon})(0)$
3: \hspace{1cm} return $q(D) + Y$
4: end function

Notice that by the properties of the Laplace distribution we have that $\text{LAPMECH}(D, q, \epsilon)$ and $\text{Lap}(\frac{\Delta q}{\epsilon})(D)$ are the same distribution, that is we can see the Laplace mechanism as returning a Laplace distribution centered in $q(D)$ with scale $\frac{\Delta q}{\epsilon}$. The scale $\frac{\Delta q}{\epsilon}$ is such that the noise that the mechanism add is directly proportional to the global sensitivity of $q$ and inversely proportional to the level of protection one wants to guarantee. Notice also that the Laplace mechanism is generic in the kind of function it takes in input, i.e. it can be applied to any numeric function, not only counting queries.

Likewise what we did for Randomized Response, we want to prove two properties of the Laplace mechanism: that it ensures differential privacy and that it has a non-trivial accuracy. Let’s start by proving that it ensures differential privacy.

Theorem 1.4  (Privacy of the Laplace mechanism)

The Laplace mechanism ensures differential privacy.

Proof. Consider $D \geq 1 \overset{\$}{\leftarrow} \Omega\{X\}^n$, $q : \mathbb{R}^{|X|} \rightarrow \mathbb{R}$, and let $p$ and $p' \overset{\$}{\leftarrow} \text{LapMech}(D, q, \epsilon)$ denote the probability density function of $\text{LapMech}(D, q, \epsilon)$ and $\text{LapMech}(D', q, \epsilon)$, respectively. We compare them at an arbitrary point $z \overset{\$}{\leftarrow} \mathbb{R}$. We have:

$$p(z) = \exp\left(-\frac{\|q(D) - z\|}{\epsilon}\right) \leq \exp\left(-\frac{\|q(D') - q(D)\|}{\epsilon}\right) \cdot \exp\left(-\frac{\|q(D) - z\|}{\epsilon}\right) = p'(z') \cdot \exp\left(-\frac{\|q(D') - q(D)\|}{\epsilon}\right).$$

Similarly, we can prove that $\exp\left(-\frac{\|q(D') - q(D)\|}{\epsilon}\right) \leq p(z) \cdot \exp\left(-\frac{\|q(D') - q(D)\|}{\epsilon}\right)$, and this concludes the proof.

Figure 1.2 gives a graphical intuition of the privacy proof. If we assume that $q$ is $c$-sensitive and we consider $q(D)$ and $q(D')$ we know that they differ for at most $c$. By adding to both of them noise according to the Laplace distribution with scale $\frac{\Delta q}{\epsilon}$ we obtain two distributions whose means are at most at distance $c$, and whose shape is given by the Laplace distribution, as depicted in Figure 1.2. Notice that the scale of the two distribution is independent from their mean and it is equal for both of them. Two such Laplace distributions have the property that for each point $z$ the ratio of their pdf evaluated in $z$ lies in the interval $[\exp(-c), \exp(c)]$. 

Pr

\begin{tikzpicture}
    \begin{axis}[
        width=\textwidth,
        height=0.5\textwidth,
        xlabel={$q(\cdot)$},
        ylabel={Pr},
        xmin=-5, xmax=5,
        ymin=0, ymax=1,
        xtick={-5,-4,-3,-2,-1,1,2,3,4,5},
        ytick={0,1},
        xticklabels={$-5$,$-4$,$-3$,$-2$,$-1$,$1$,$2$,$3$,$4$,$5$},
        yticklabels={0,1},
        axis lines=left,
        legend pos=north west,
    ]
    \addplot[blue, domain=-5:5, samples=100] {exp(-\x^2/2)}/\x/2*pi;\addlegendentry{c=1}
    \addplot[red, domain=-5:5, samples=100] {exp(-\x^2/2)}/\x/2*pi;\addlegendentry{c=2}
    \end{axis}
\end{tikzpicture}
Laplace Mechanism

**Accuracy Theorem:** let \( r = \text{LapMech}(D, q, \epsilon) \)

\[
\Pr \left[ |q(D) - r| \geq \left( \frac{\Delta q}{\epsilon} \right) \ln \left( \frac{1}{\beta} \right) \right] = \beta
\]
Local sensitivity

**Definition 1.8** (Global sensitivity). The *global sensitivity* of a function $q : \mathcal{X}^n \rightarrow \mathbb{R}$ is:

$$\Delta q = \max \left\{ |q(D) - q(D')| \mid D \sim_1 D' \in \mathcal{X}^n \right\}$$

**Definition 1.14** (Local sensitivity). The *local sensitivity* of a function $q : \mathcal{X}^n \rightarrow \mathbb{R}$ at $D \in \mathcal{X}^n$ is:

$$\ell \Delta q(D) = \max \left\{ |q(D) - q(D')| \mid D \sim_1 D', D' \in \mathcal{X}^n \right\}$$
Calibrating noise to the local sensitivity

We may add noise proportional to the local sensitivity (LS).

Unfortunately, this does not guarantee privacy.

Suppose that for a given D we have LS(D)=0 but that we also have D~D’ with LS(D’)=10^9.
Calibrating noise to the local sensitivity

We may add noise proportional to the local sensitivity (LS).

Unfortunately, this does not guarantee privacy.

Suppose that for a given $D$ we have $LS(D)=0$ but that we also have $D \sim D'$ with $LS(D')=10^9$.

We will see that we can do anyway better than GS.
Smooth Sensitivity

**Definition 2.2** (Smooth sensitivity). For $\beta > 0$, the $\beta$-smooth sensitivity of $f$ is

$$S_{f,\beta}^*(x) = \max_{y \in \mathcal{D}^n} \left( LS_f(y) \cdot e^{-\beta d(x,y)} \right).$$

[Nissim, Raskhodnikova, Smith '06]
Smooth Sensitivity

**Definition 2.1 (A Smooth Bound on \( LS \)).** For \( \beta > 0 \), a function \( S : D^n \to \mathbb{R}^+ \) is a \( \beta \)-smooth upper bound on the local sensitivity of \( f \) if it satisfies the following requirements:

\[
\forall x \in D^n : \quad S(x) \geq LS_f(x) ; \\
\forall x, y \in D^n, d(x, y) = 1 : \quad S(x) \leq e^{\beta} \cdot S(y) .
\]

**Definition 2.2 (Smooth sensitivity).** For \( \beta > 0 \), the \( \beta \)-smooth sensitivity of \( f \) is

\[
S_{f, \beta}^*(x) = \max_{y \in D^n} \left( LS_f(y) \cdot e^{-\beta d(x, y)} \right).
\]

[Nissim, Raskhodnikova, Smith '06]
Lemma 2.6. Let $h$ be an $(\alpha, \beta)$-admissible noise probability density function, and let $Z$ be a fresh random variable sampled according to $h$. For a function $f : D^n \rightarrow \mathbb{R}^d$, let $S : D^n \rightarrow \mathbb{R}$ be a $\beta$-smooth upper bound on the local sensitivity of $f$. Then algorithm $A(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$ is $(\epsilon, \delta)$-differentially private.

For two neighbor databases $x$ and $y$, the output distribution $A(y)$ is a shifted and scaled version of $A(x)$. The sliding and dilation properties ensure that $\Pr[A(x) \in S]$ and $\Pr[A(y) \in S]$ are close for all sets $S$ of outputs.

[Nissim, Raskhodnikova, Smith ’06]
Admissible Noise

Adding noise $O(\text{SS}^\varepsilon_q(x)/\varepsilon)$ (according to a Cauchy distribution) is sufficient for $\varepsilon$-differential privacy.

Laplace and Gauss give $(\varepsilon, \delta)$-DP

[Nissim, Raskhodnikova, Smith '06]
Admissible Noise

Adding noise $O(\text{SS}_q^\varepsilon(x)/\varepsilon)$ (according to a Cauchy distribution) is sufficient for $\varepsilon$-differential privacy.

Laplace and Gauss give $(\varepsilon,\delta)$-DP

Computing the Smooth Sensitivity can be intractable.

[Nissim, Raskhodnikova, Smith '06]
Accuracy revisited

Accuracy Theorem (smooth sensitivity using Laplace):

\[ ||q(D) - r||_\infty \in O\left(\frac{S(D)}{\epsilon}\right) \]

Where

\[ ||\vec{v}||_\infty = \max_{i=0}^{\infty} |v_i| \]
Propose-test-release

Given \( q : \mathcal{X}^n \rightarrow \mathbb{R}, \epsilon, \delta, \beta \geq 0 \)

1. Propose a target bound \( \beta \) on local sensitivity.

2. Let \( \hat{d} = d(x, \{ x' : \text{LS}_q(x') > \beta \}) + \text{Lap}(1/\epsilon) \), where \( d \) denotes Hamming distance.

3. If \( \hat{d} \leq \ln(1/\delta)/\epsilon \), output \( \perp \).

4. If \( \hat{d} > \ln(1/\delta)/\epsilon \), output \( q(x) + \text{Lap}(\beta/\epsilon) \).
**Stability-based algorithms**

---

**Releasing stable values**

Given \( q : \mathcal{X}^n \rightarrow \mathbb{R}, \epsilon, \delta \geq 0 \)

1. Let \( \hat{d} = d(x, \{x' : q(x') \neq q(x)\}) + \text{Lap}(1/\epsilon) \), where \( d \) denotes Hamming distance.
2. If \( \hat{d} \leq 1 + \ln(1/\delta)/\epsilon \), output \( \bot \).
3. Otherwise output \( q(x) \).
Stability-based algorithms

Releasing stable values Given $q : X^n \to \mathbb{R}$, $\epsilon, \delta \geq 0$

1. Let $\hat{d} = d(x, \{x' : q(x') \neq q(x)\}) + \text{Lap}(1/\epsilon)$, where $d$ denotes Hamming distance.
2. If $\hat{d} \leq 1 + \ln(1/\delta)/\epsilon$, output $\perp$.
3. Otherwise output $q(x)$.

Proposition 3.3 (releasing stable values). For every query $q : X^n \to Y$ and $\epsilon, \delta > 0$, the above algorithm is $(\epsilon, \delta)$-differentially private.
Stability-based algorithms
Stability-based algorithms

Consider, for example, the *mode* function \(q : \mathcal{X}^n \rightarrow \mathcal{X}\), where \(q(x)\) is defined to be the most frequently occurring data item in \(x\) (breaking ties arbitrarily). Then \(d(x, \{x' : q(x') \neq q(x)\})\) equals half of the gap in the number of occurrences between the mode and the second-most frequently occurring item (rounded up). So we have:

**Proposition 3.4** (stability-based mode). For every data universe \(\mathcal{X}\), \(n \in \mathbb{N}\), \(\varepsilon, \delta \geq 0\), there is an \((\varepsilon, \delta)\)-differentially private algorithm \(\mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{X}\) such that for every dataset \(x \in \mathcal{X}^n\) where the difference between the number of occurrences of the mode and the 2nd most frequently occurring item is larger than \(4[\ln(1/\delta)/\varepsilon]\), \(\mathcal{M}(x)\) outputs the mode of \(x\) with probability at least \(1 - \delta\).
Stability-based Histogram

1. For every point \( y \in X \):

   (a) If \( q_y(x) = 0 \), then set \( a_y = 0 \).

   (b) If \( q_y(x) > 0 \), then:

      i. Set \( a_y \leftarrow q_y(x) + \text{Lap}(2/\epsilon n) \).

      ii. If \( a_y < 2 \ln(2/\delta)/\epsilon n + 1/n \), then set \( a_y \leftarrow 0 \).

2. Output \((a_y)_{y \in X}\).
1. For every point $y \in \mathcal{X}$:
   
   (a) If $q_y(x) = 0$, then set $a_y = 0$.
   
   (b) If $q_y(x) > 0$, then:
       
       i. Set $a_y \leftarrow q_y(x) + \text{Lap}(2/\varepsilon n)$.
       
       ii. If $a_y < 2 \ln(2/\delta)/\varepsilon n + 1/n$, then set $a_y \leftarrow 0$.

2. Output $(a_y)_{y \in \mathcal{X}}$. 

---

**Stability-based Histogram**
Utility: The algorithm gives exact answers for queries $q_y$ where $q_y(x) = 0$. There are at most $n$ queries $q_y$ with $q_y(x) > 0$ (namely, ones where $y \in \{x_1, \ldots, x_n\}$). By the tails of the Laplace distribution and a union bound, with high probability all of the noisy answers $q_y(x) + \text{Lap}(2/\varepsilon n)$ computed in Step 1(b) have error at most $O((\log n)/\varepsilon n) \leq O(\log(1/\delta)/\varepsilon n)$. Truncating the small values to zero in Step 1(b) introduces an additional error of up to $2\ln(1/\delta)/\varepsilon n + 1/n = O(\log(1/\delta)/\varepsilon n)$. 
Utility: The algorithm gives exact answers for queries $q_y$ where $q_y(x) = 0$. There are at most $n$ queries $q_y$ with $q_y(x) > 0$ (namely, ones where $y \in \{x_1, \ldots, x_n\}$). By the tails of the Laplace distribution and a union bound, with high probability all of the noisy answers $q_y(x) + \text{Lap}(2/\varepsilon n)$ computed in Step 1(b)i have error at most $O((\log n)/\varepsilon n) \leq O(\log(1/\delta)/\varepsilon n)$. Truncating the small values to zero in Step 1(b)ii introduces an additional error of up to $2\ln(1/\delta)/\varepsilon n + 1/n = O(\log(1/\delta)/\varepsilon n)$. 

Stability-based Histogram
Accuracy with the standard histogram DP algorithm:

$$|q_h(D) - r_h| \leq O \left( \frac{\log(|X|)}{n} \right)$$

Accuracy with the stable histogram DP algorithm:

$$|q_h(D) - r_h| \leq O \left( \frac{\log(1/\delta)}{n} \right)$$
How can we make this reasoning mathematically precise?
Formal Semantics

We need to assign a formal meaning to the different components:

- **Precondition**
- **Program**
- **Postcondition**
Formal Semantics

We need to assign a formal meaning to the different components:

- Precondition
- Program
- Postcondition

formal semantics of programs
We need to assign a formal meaning to the different components:

- Precondition
- Program
- Postcondition

**formal semantics of specification conditions**

**formal semantics of programs**

**formal semantics of specification conditions**
Formal Semantics

We need to assign a formal meaning to the different components:

- **Precondition**
- **Program**
- **Postcondition**

We also need to describe the rules which combine program and specifications.

- Formal semantics of specification conditions
- Formal semantics of programs
- Formal semantics of specification conditions
An example

\begin{verbatim}
FastExponentiation(n, k : Nat) : Nat
n' := n; k' := k; r := 1;
if k' > 0 then
    while k' > 1 do
        if even(k') then
            n' := n' * n';
            k' := k' / 2;
        else
            r := n' * r;
            n' := n' * n';
            k' := (k' - 1) / 2;
        r := n' * r;
(* result is r *)
\end{verbatim}
Programming Language

c ::= abort
    | skip
    | x := e
    | c ; c
    | if e then c else c
    | while e do c

x, y, z, ... program variables

\( e_1, e_2, \ldots \) expressions

\( c_1, c_2, \ldots \) commands
Expressions

We want to be able to write complex programs with our language.

\[
e ::= x \quad | \quad f(e_1, \ldots, e_n)
\]

Where \( f \) can be any arbitrary operator.

Some expression examples

\[
\begin{align*}
x+5 & \quad x \mod k & \quad x[i] & \quad (x[i+1] \mod 4)+5
\end{align*}
\]
Memories

We can formalize a memory as a map $m$ from variables to values.

$$m = [x_1 \rightarrow v_1, \ldots, x_n \rightarrow v_n]$$

We consider only maps that respect types.
Memories

We can formalize a memory as a map $m$ from variables to values.

$$m = [x_1 \mapsto v_1, \ldots, x_n \mapsto v_n]$$

We consider only maps that respect types.

We want to read the value associated to a particular variable:

$$m(x)$$

We want to update the value associated to a particular variable:

$$m[x \leftarrow v]$$

This is defined as

$$m[x \leftarrow v](y) = \begin{cases} v & \text{if } x = y \\ m(y) & \text{otherwise} \end{cases}$$
Semantics of Expressions

What is the meaning of the following expressions?

\[ x + 5 \quad x \mod k \quad x[i] \quad (x[i+1] \mod 4) + 5 \]
Semantics of Expressions

What is the meaning of the following expressions?

\[ x+5 \quad x \mod k \quad x[i] \quad (x[i+1] \mod 4)+5 \]

We can give the semantics as a relation between expressions, memories and values.

\[ \text{Exp} \ast \text{Mem} \ast \text{Val} \]

We will denote this relation as:

\[ \{e\}_m = v \]
Semantics of Expressions

What is the meaning of the following expressions?

\[ x + 5 \quad x \mod k \quad x[i] \quad (x[i+1] \mod 4)+5 \]

We can give the semantics as a relation between expressions, memories and values.

\[ \text{Exp} * \text{Mem} * \text{Val} \]

We will denote this relation as:

\[ \{ e \}_m = v \]

This is commonly typeset as:

\[ [e]_m = v \]
Semantics of Commands

What is the meaning of the following command?

\[ k := 2; \; z := x \mod k; \; \text{if } z = 0 \text{ then } r := 1 \text{ else } r := 2 \]
Semantics of Commands

What is the meaning of the following command?

\[
\text{k:=2; } \text{z:=x mod k; if } \text{z=0 then r:=1 else r:=2}
\]

We can give the semantics as a relation between command, memories and memories or failure.

\[
\text{Exp } \ast \text{ Mem } \ast (\text{Mem } \mid \perp)
\]

We will denote this relation as:

\[
\{ c \}_m = m' \quad \text{Or} \quad \{ c \}_m = \perp
\]
Semantics of Commands

What is the meaning of the following command?

\[ k := 2; \ z := x \mod k; \ \text{if } z = 0 \text{ then } r := 1 \ \text{else } r := 2 \]

We can give the semantics as a relation between \textit{command}, \textit{memories} and \textit{memories} or failure.

\[ \text{Exp} \times \text{Mem} \times (\text{Mem} \mid \bot) \]

We will denote this relation as:

\[ \{ c \}_m = m' \quad \text{Or} \quad \{ c \}_m = \bot \]

This is commonly typeset as:

\[ [c]_m = m' \]
Semantics of Commands

This is defined on the structure of commands:

\[
\begin{align*}
\{\text{abort}\}_m &= \bot \\
\{\text{skip}\}_m &= m \\
\{x:=e\}_m &= m[x \leftarrow \{e\}_m] \\
\{c;c'\}_m &= \{c'\}_m', \\
\{c;c'\}_m &= \bot, \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \begin{cases} 
\{c_t\}_m & \text{if } \{e\}_m = \text{true} \\
\{c_f\}_m & \text{if } \{e\}_m = \text{false} 
\end{cases} \\
\{\text{while } e \text{ do } c\}_m &= \sup_{n \in \text{Nat}} \{\text{while}_n e \text{ do } c\}_m
\end{align*}
\]

where

\[
\text{while}_n e \text{ do } c = \text{while}_n e \text{ do } c; \text{if } e \text{ then } \text{abort} \text{ else } \text{skip}
\]

and
Semantics of Commands

This is defined on the structure of commands:

\[
\begin{align*}
\{\text{abort}\}_m &= \bot \\
\{\text{skip}\}_m &= m \\
\{x:=e\}_m &= m[x\leftarrow\{e\}_m] \\
\{c;c'\}_m &= \{c'\}_m, \quad \text{if} \quad \{c\}_m = m' \\
\{c;c'\}_m &= \bot, \quad \text{if} \quad \{c\}_m = \bot \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \{c_t\}_m, \quad \text{if } \{e\}_m=\text{true} \\
\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m &= \{c_f\}_m, \quad \text{if } \{e\}_m=\text{false} \\
\{\text{while } e \text{ do } c\}_m &= \sup_{n \in \mathbb{N}} \{\text{while}_n e \text{ do } c\}_m
\end{align*}
\]

where

\[
\text{while}_n e \text{ do } c = \text{while}_{n+1} e \text{ do } c; \text{if } e \text{ then abort else skip}
\]

and

\[
\text{while}_0 e \text{ do } c = \text{skip} \\
\text{while}_{n+1} e \text{ do } c = \text{if } e \text{ then } (c; \text{while}_n e \text{ do } c) \text{ else skip}
\]
Formal Semantics

We need to assign a formal meaning to the different components:

- Precondition
- Program
- Postcondition
Formal Semantics

We need to assign a formal meaning to the different components:

Precondition
Program
Postcondition

formal semantics of programs
Formal Semantics

We need to assign a formal meaning to the different components:
We need to assign a formal meaning to the different components:

- **Precondition**
- **Program**
- **Postcondition**

We also need to describe the rules which combine program and specifications.

**Formal Semantics**

**formal semantics of programs**

**formal semantics of specification conditions**

**formal semantics of specification conditions**
Some examples

\[
i := 0;
\]
\[
r := 1;
\]
\[
\text{while}(i \leq k) \text{do}
\]
\[
r := r \times n;
\]
\[
i := i + 1
\]

Precondition
\[
\{0 < k\} \Rightarrow \{r = n^k\}
\]

Postcondition

Is it a good specification?
Some examples

i := 0;
r := 1;
while (i ≤ k) do
    r := r * n;
i := i + 1

Precondition

: \{0 < k\} \Rightarrow \{r = n^k\}

Postcondition

Is it a good specification? ✗
Some examples

```
i := 0;
r := 1;
while (i ≤ k) do
  r := r * n;
i := i + 1
```

Precondition
\[
\{0 < k\} \Rightarrow \{ r = n^k \}
\]

Postcondition

Is it a good specification?

\[
m_{in} = [k = 1, n = 2, i = 0, r = 0]
m_{out} = [k = 1, n = 2, i = 2, r = 4]
\]
Some examples

i := 0;
r := 1;
while (i < k) do
  r := r * n;
i := i + 1

Precondition
\[ \{0 \leq k\} \Rightarrow \{r = n^k\}\]

Postcondition

Is it a good specification?
Some examples

\[ i := 0; \]
\[ r := 1; \]
while (i < k) do
  \[ r := r \times n; \]
  \[ i := i + 1 \]

Precondition
\[ \{ 0 \leq k \} \Rightarrow \{ r = n^k \} \]

Postcondition

Is it a good specification?
Some examples

\[
i := 0; \\
r := 1; \\
\text{while}(i \leq k) \text{do} \\
\quad r := r \times n; \\
\quad i := i + 1
\]

Precondition

\[
\{ 0 \leq k \} \Rightarrow \{ r = n^i \}
\]

Postcondition

Is it a good specification?
Some examples

Precondition

\begin{align*}
i &:= 0; \\
r &:= 1; \\
\text{while}(i \leq k) \text{do} & \\
& \quad r := r \times n; \\
i &:= i + 1
\end{align*}

\Rightarrow \{ r = n^i \}

Postcondition

Is it a good specification?  

✓
How do we determine the validity of an Hoare triple?
Validity of Hoare triple

Precondition (a logical formula)

\[ c : P \Rightarrow Q \]

Program

Postcondition (a logical formula)
Validity of Hoare triple

We are interested only in inputs that meets $P$ and we want to have outputs satisfying $Q$.

$c : P \Rightarrow Q$
Validity of Hoare triple

\[ c : P \Rightarrow Q \]

Precondition (a logical formula)

Postcondition (a logical formula)

We are interested only in inputs that meets P and we want to have outputs satisfying Q.

How shall we formalize this intuition?
Validity of Hoare triple

We say that the triple $c : P \Rightarrow Q$ is valid if and only if for every memory $m$ such that $P(m)$ and memory $m'$ such that $\{c\}_{m=m'}$ we have $Q(m')$. 
Validity of Hoare triple
We say that the triple $c : P \Rightarrow Q$ is valid if and only if for every memory $m$ such that $P(m)$ and memory $m'$ such that $\{c\}_m = m'$ we have $Q(m')$.

Is this condition easy to check?
Rules of Hoare Logic:

\[\vdash \text{skip} : P \Rightarrow P\]

\[\vdash x := e : P[e/x] \Rightarrow P\]

\[\vdash c; c' : P \Rightarrow Q\]

\[\vdash c : P \Rightarrow R\]
\[\vdash c' : R \Rightarrow Q\]

\[\vdash c; c' : P \Rightarrow Q\]

\[\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q\]

\[\vdash c : e \land P \Rightarrow P\]

\[\vdash \text{while } e \text{ do } c : P \Rightarrow P \land \neg e\]
Rules of Hoare Logic:

\[ \vdash \text{skip: } P \Rightarrow P \]

\[ \vdash x := e : P[e/x] \Rightarrow P \]

\[ \vdash c ; c' : P \Rightarrow Q \]

\[ \vdash c : P \Rightarrow R \]
\[ \vdash c' : R \Rightarrow Q \]
\[ \vdash c ; c' : P \Rightarrow Q \]

\[ \vdash c_1 : e \land P \Rightarrow Q \]
\[ \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q \]

\[ \vdash c : e \land P \Rightarrow P \]
\[ \vdash \text{while } e \text{ do } c : P \Rightarrow P \land \neg e \]
Rules of Hoare Logic:

\[ \vdash \text{skip} : P \Rightarrow P \]

\[ \vdash x := e : P[e/x] \Rightarrow P \]

\[ \vdash c; c' : P \Rightarrow Q \]

\[ \vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q \quad \vdash c; c' : P \Rightarrow Q \]

\[ \vdash c : P \Rightarrow Q \]

\[ \vdash c_1 : e \land P \Rightarrow Q \quad \vdash c_2 : \neg e \land P \Rightarrow Q \quad \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q \]

\[ \vdash c : e \land P \Rightarrow P \]

\[ \vdash \text{while } e \text{ do } c : P \Rightarrow P \land \neg e \]
Some examples

⊢ \begin{align*} & x := z \times 2; z := x \times 2 \\
& : \{ z \times 4 = 8 \} \Rightarrow \{ z = 8 \} \end{align*}

Is this a valid triple?
Some examples

\[ \vdash x := z \times 2; z := x \times 2 \]

\[ : \{ z \times 4 = 8 \} \Rightarrow \{ z = 8 \} \]

Is this a valid triple? √
Some examples

\[
\vdash x := z \ast 2; z := x \ast 2 \\
\vdash \{z \ast 4 = 8\} \Rightarrow \{z = 8\}
\]

Is this a valid triple?  

Can we prove it with the rules that we have?
Some examples

\[ \vdash x := z \ast 2; z := x \ast 2 \]

: \{ z \ast 4 = 8 \} \Rightarrow \{ z = 8 \}

Is this a valid triple? ✓

Can we prove it with the rules that we have? ✓
Some Examples

\[ \vdash x := z \cdot 2 \ (\{ z \cdot 2 \} \cdot 2 = 8) \Rightarrow \{ x \cdot 2 = 8 \} \]

\[ \{ z \cdot 4 = 8 \} \Rightarrow \{ (z \cdot 2) \cdot 2 = 8 \} \]

\[ \vdash x := z \cdot 2; \ z := x \cdot 2: \ \{ z \cdot 4 = 8 \} \Rightarrow \{ z = 8 \} \]
Soundness

If we can derive $\vdash c : P \implies Q$ through the rules of the logic, then the triple $c : P \implies Q$ is valid.
Relative Completeness

\[ P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q \]

\[ \vdash c : P \Rightarrow Q \]
Relative Completeness

\[ P \Rightarrow S \quad \vdash \quad c : S \Rightarrow R \quad R \Rightarrow Q \]

\[ \vdash \quad c : \quad P \Rightarrow Q \]

If a triple \( c : \text{Pre} \Rightarrow \text{Post} \) is valid, and we have an oracle to derive all the true statements of the form \( P \Rightarrow S \) and of the form \( R \Rightarrow Q \), which we can use in applications of the conseq rule, then we can derive \( \vdash c : \text{Pre} \Rightarrow \text{Post} \) through the rules of the logic.
A logic for information flow control
Private vs Public

We want to distinguish confidential information that need to be kept secret from nonconfidential information that can be accessed by everyone.

We assume that every variable is tagged with one either public or private.

\[ x: \text{public} \quad x: \text{private} \]
We want to guarantee that confidential information do not flow in what is considered nonconfidential.
Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.
Is this program secure?

\[
\begin{aligned}
x & : \text{private} \\
y & : \text{public} \\
x & := y
\end{aligned}
\]
Is this program secure?

\[
\begin{align*}
x : \text{private} \\
y : \text{public} \\
x := y
\end{align*}
\]

Secure
Is this program secure?

\[
\begin{array}{c}
x: \text{private} \\
y: \text{public}
\end{array}
\]

\[
y := x
\]
Is this program secure?

\[
\begin{array}{c}
x: \text{private} \\
y: \text{public} \\
y := x
\end{array}
\]

Insecure
Is this program secure?

\[
x: \text{private} \\
y: \text{public} \\
y := x; \\
y := 5
\]
Is this program secure?

```
x:private
y:public

y := x;
y := 5

Secure
```
Is this program secure?

x:.private
y:.public

if y mod 3 = 0 then
  x:=1
else
  x:=0
Is this program secure?

x: private
y: public

if y mod 3 = 0 then
    x := 1
else
    x := 0

Secure
Is this program secure?

\[
\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
\text{if } x \mod 3 = 0 \text{ then} & \\
\quad y &= 1 \\
\text{else} & \\
\quad y &= 0
\end{align*}
\]
Is this program secure?

\[
x: \text{private} \\
y: \text{public}
\]

if \(x \mod 3 = 0\) then
  \(y := 1\)
else
  \(y := 0\)

Insecure
How can we formulate a policy that forbids flows from private to public?
Low equivalence

Two memories $m_1$ and $m_2$ are low equivalent if and only if they coincide in the value that they assign to public variables.

In symbols: $m_1 \sim_{\text{low}} m_2$
Noninterference

In symbols

\( m_1 \sim_{\text{low}} m_2 \) and \( \{c\}_{m_1}=m_1' \) and \( \{c\}_{m_2}=m_2' \)

implies \( m_1' \sim_{\text{low}} m_2' \)
Does this program satisfy noninterference?

\[
\begin{align*}
\text{x:private} \\
\text{y:public} \\
\text{x:=y}
\end{align*}
\]
Does this program satisfy noninterference?

\[
x \text{: private} \\
y \text{: public} \\
x := y
\]

Yes
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
x := y
\]

\[\text{min}_1 = [x = n_1, y = k]\]

Yes
Does this program satisfy noninterference?

\begin{align*}
\text{x:private} \\
\text{y:public} \\
\text{x:=y}
\end{align*}

\begin{align*}
\text{m^{in}_1} &= [x=n_1, y=k] \\
\text{m^{in}_2} &= [x=n_2, y=k]
\end{align*}

Yes
Does this program satisfy noninterference?

\[
\begin{align*}
  &x : \text{private} \\
  &y : \text{public} \\
  &x := y
\end{align*}
\]

\[
\begin{align*}
  m^{\text{in}}_1 &= [x=n_1, y=k] & m^{\text{in}}_2 &= [x=n_2, y=k] \\
  m^{\text{out}}_1 &= [x=k, y=k] & m^{\text{out}}_2 &= [x=k, y=k]
\end{align*}
\]

Yes
Does this program satisfy noninterference?

\[
x : \text{private} \\
y : \text{public} \\
y := x
\]
Does this program satisfy noninterference?

\[
x \colon \text{private} \\
y \colon \text{public} \\
y := x
\]  

No
Does this program satisfy noninterference?

\[
\begin{array}{ll}
  x : \text{private} \\
  y : \text{public} \\
  y := x
\end{array}
\]

\[m_{in_1} = [x=n_1, y=k]\]
Does this program satisfy noninterference?

\[ \text{x:private} \]
\[ \text{y:public} \]
\[ y := x \]

\[ m_{\text{in}_1} = [x=n_1, y=k] \]
\[ m_{\text{in}_2} = [x=n_2, y=k] \]
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
y := x
\]

\[
m_{\text{in}1} = [x=n_1, y=k] \\
m_{\text{out}1} = [x=n_1, y=n_1]
\]

\[
m_{\text{in}2} = [x=n_2, y=k] \\
m_{\text{out}2} = [x=n_2, y=n_2]
\]

\textbf{No}
Does this program satisfy noninterference?

```
x: private
y: public

y := x
y := 5
```
Does this program satisfy noninterference?

x: private
y: public

y := x
y := 5

Yes
Does this program satisfy noninterference?

\[
x\text{::private} \\
y\text{::public} \\
y:=x \\
y:=5
\]

\[m^{in_1}=[x=n_1,y=k]\]

Yes
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public}
\]

\[
y := x \\
y := 5
\]

\[m^{in_1} = [x=n_1, y=k]\]

\[m^{in_2} = [x=n_2, y=k]\]

Yes
Does this program satisfy noninterference?

\[
\begin{array}{c}
x: \text{private} \\
y: \text{public}
\end{array}
\]

\[
\begin{align*}
y & := x \\
y & := 5
\end{align*}
\]

\[
\begin{array}{ll}
m_{in_1} = [x=n_1, y=k] & \quad m_{in_2} = [x=n_2, y=k] \\
m_{out_1} = [x=n_1, y=5] & \quad m_{out_2} = [x=n_2, y=5]
\end{array}
\]

Yes
Does this program

x: private
y: public
if y mod 3 = 0 then
  x := 1
else
  x := 0
Does this program

\begin{verbatim}
x:private
y:public
if y mod 3 = 0 then
  x:=1
else
  x:=0
\end{verbatim}

Yes
Does this program

\[
x: \text{private} \\
y: \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} \\
\quad x := 1 \\
\text{else} \\
\quad x := 0
\]

\[m_{in_1} = [x=n_1, y=6]\]
Does this program

```
x: private
y: public
if y mod 3 = 0 then
  x := 1
else
  x := 0
```

$m^{in_1} = [x = n_1, y = 6]$  \hspace{1cm}  $m^{in_2} = [x = n_2, y = 6]$
Does this program

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} & \\
\quad x & := 1 \\
\text{else} & \\
\quad x & := 0
\end{align*}
\]

\[
\begin{align*}
m_{\text{in}1} & = [x=n_1, y=6] & m_{\text{in}2} & = [x=n_2, y=6] \\
m_{\text{out}1} & = [x=1, y=6] & m_{\text{out}2} & = [x=1, y=6]
\end{align*}
\]
Does this program

```plaintext
x:private
y:public
if x mod 3 = 0 then
    y:=1
else
    y:=0
```
Does this program

x: private
y: public
if x mod 3 = 0 then
  y := 1
else
  y := 0

No
Does this program

\[
x: \text{private} \\
y: \text{public} \\
\text{if } x \mod 3 = 0 \text{ then} \\
\quad y := 1 \\
\text{else} \\
\quad y := 0 \\
\]

\[m_{in_1} = [x=6, y=k]\]
Does this program

\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
\text{if } x \mod 3 &= 0 \text{ then} \\
& \quad y := 1 \\
\text{else} \\
& \quad y := 0
\end{align*}

\text{No}

m_{in_1} = [x=6, y=k] \quad m_{in_2} = [x=5, y=k]
Does this program

\[
x: \text{private} \\
y: \text{public} \\
\text{if } x \mod 3 = 0 \text{ then } \\
\quad y := 1 \\
\text{else} \\
\quad y := 0
\]

\[
m_{\text{in}1} = [x=6, y=k] \\
m_{\text{out}1} = [x=6, y=1] \\
m_{\text{in}2} = [x=5, y=k] \\
m_{\text{out}2} = [x=5, y=0]
\]

No
Does this program

s1: public
s2: private
r: private
i: public

proc Compare (s1: list[n] bool, s2: list[n] bool)
i := 0;
r := 0;
while i < n \ r = 0 do
  if not(s1[i] = s2[i]) then
    r := 1
  i := i + 1
Does this program

s1: public
s2: private
r: private
i: public

proc Compare (s1:list[n] bool, s2:list[n] bool)
i:=0;
r:=0;
while i<n /\ r=0 do
  if not(s1[i]=s2[i]) then
    r:=1
  i:=i+1

No
How can we prove our programs noninterferent?
Can we use the tool we studied so far?

c : \( P \Rightarrow Q \)
Validity of Hoare triple

We say that the triple $c : P \Rightarrow Q$ is valid if and only if for every memory $m$ such that $P(m)$ and memory $m'$ such that $\{c\}_{m=m'}$ we have $Q(m')$. 
Validity of Hoare triple
We say that the triple \( c : P \Rightarrow Q \) is valid if and only if for every memory \( m \) such that \( P(m) \) and memory \( m' \) such that \( \{c\}_m = m' \) we have \( Q(m') \).

Validity talks only about one memory. How can we manage two memories?
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$, $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$.
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$, $\{c\}_{m_1}=m_1'$ and $\{c\}_{m_2}=m_2'$ implies $m_1' \sim_{\text{low}} m_2'$
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$, \{c\}_{m_1}=m_1'$ and \{c\}_{m_2}=m_2'$ implies $m_1' \sim_{\text{low}} m_2'$
Relational Property

In symbols, $c$ is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$, $\{c\}_{m_1=\text{m}_1'}$ and $\{c\}_{m_2=\text{m}_2'}$ implies $m_1' \sim_{\text{low}} m_2'$. 

\[ \\
\text{private} \quad \text{public} \quad \text{private} \quad \text{public} \\
V \quad C \quad C \quad V \\
\text{private} \quad \text{public} \quad \text{private} \quad \text{public} \\
\text{V} \quad \text{V} \quad \text{V} \quad \text{V} \\
\]
In symbols, $c$ is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$, $\{c\}_{m_1}=m_1'$ and $\{c\}_{m_2}=m_2'$ implies $m_1' \sim_{\text{low}} m_2'$. 
Relational Property

In symbols, c is noninterferent if and only if for every \( m_1 \sim_{\text{low}} m_2, \{c\}_{m_1}=m_1' \) and \( \{c\}_{m_2}=m_2' \) implies \( m_1' \sim_{\text{low}} m_2' \).
Relational Property

In symbols, \( c \) is noninterferent if and only if for every \( m_1 \sim_{\text{low}} m_2, \{c\}_{m_1} = m'_1 \) and \( \{c\}_{m_2} = m'_2 \) implies \( m'_1 \sim_{\text{low}} m'_2 \).
Relational Hoare Logic - RHL

\[ c_1 \sim c_2 : P \Rightarrow Q \]

Precondition (a logical formula)

Program \( \sim \) Program

Postcondition (a logical formula)
Relational Assertions

$c_1 \sim c_2 : P \Rightarrow Q$

Need to talk about variables of the two memories
Need to talk about variables of the two memories

$c_1 \sim c_2 : x\langle 1 \rangle \leq x\langle 2 \rangle \Rightarrow x\langle 1 \rangle \geq x\langle 2 \rangle$
Relational Assertions

$c_1 \sim c_2 : P \Rightarrow Q$

- Need to talk about variables of the two memories

$c_1 \sim c_2 : x(1) \leq x(2) \Rightarrow x(1) \geq x(2)$

- Tags describing which memory we are referring to.
Rules of Relational Hoare Logic

Skip

\[ \vdash \text{skip} \Rightarrow \text{P} \]
Rules of Relational Hoare Logic Composition

\[ \vdash c_1 \sim c_2 : P \Rightarrow R \quad \vdash c_1' \sim c_2' : R \Rightarrow S \]

\[ \vdash c_1 ; c_1' \sim c_2 ; c_2' : P \Rightarrow S \]
We can weaken $P$, i.e. replace it by something that is implied by $P$. In this case $S$.

We can strengthen $Q$, i.e. replace it by something that implies $Q$. In this case $R$. 
Rules of Relational Hoare Logic

Assignment

\[ \vdash x_1 := e_1 \sim x_2 := e_2 : \]

\[ P[e_1<1>/x_1<1>, e_2<2>/x_2<2>] \Rightarrow P \]
Rules of Relational Hoare Logic

If then else

\[ P \Rightarrow e_1<1>=e_2<2> \]

\[ \vdash c_1\neg c_2:e_1<1>\land P \Rightarrow Q \]

\[ \vdash c_1'\neg c_2':\neg e_1<1>\land P \Rightarrow Q \]

\[ \text{if } e_1 \text{ then } c_1 \text{ else } c_1' \]

\[ \vdash \neg \text{if } e_2 \text{ then } c_2 \text{ else } c_2' : P \Rightarrow Q \]
Rules of Relational Hoare Logic
If then else - left

$$\vdash c_1 \sim c_2 : e < 1 > \land P \Rightarrow Q$$
$$\vdash c_1' \sim c_2 : \neg e < 1 > \land P \Rightarrow Q$$

if e then c₁ else c₁′
$$\vdash \sim C_2 : P \Rightarrow Q$$
Rules of Relational Hoare Logic
If then else - left

\[ \vdash c_1 \sim c_2 : e < 2 > \land P \Rightarrow Q \]
\[ \vdash c_1 \sim c_2' : \neg e < 2 > \land P \Rightarrow Q \]

\[ \vdash c_1 \sim c_2' : P \Rightarrow Q \]
Soundness

If we can derive $\vdash \neg c_1 \sim c_2 : P \Rightarrow Q$ through the rules of the logic, then the quadruple $c_1 \sim c_2 : P \Rightarrow Q$ is valid.
Relative Completeness

If a quadruple \( c_1 \sim c_2 : P \Rightarrow Q \) is valid, and we have an oracle to derive all the true statements of the form \( P \Rightarrow S \) and of the form \( R \Rightarrow Q \), then we can derive \( \vdash c_1 \sim c_2 : P \Rightarrow Q \) through the rules of the logic.
Soundness and completeness with respect to Hoare Logic

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

iff

$$\vdash_{\text{HL}} c_1 ; c_2 : P \Rightarrow Q$$
Soundness and completeness with respect to Hoare Logic

\[ \vdash_{\text{RHL}} c_1 \sim c_2 : P \Rightarrow Q \]

iff

\[ \vdash_{\text{HL}} c_1 ; c_2 : P \Rightarrow Q \]

Under the assumption that we can partition the memory adequately, and that we have termination.
Questions?