The hybrid argument
Indistinguishability of probability distributions

- For each $\eta \in \mathbb{N}$ let $D_\eta^0$ and $D_\eta^1$ be probability distributions over bit-strings.
- The families of probability distributions $D^0 = \{D_\eta^0\}_{\eta \in \mathbb{N}}$ and $D^1 = \{D_\eta^1\}_{\eta \in \mathbb{N}}$ are indistinguishable if
  - for any adversary $\mathcal{A}$
    - The running time of $\mathcal{A}(\eta, \cdot)$ must be polynomial in $\eta$
    - the difference of probabilities

\[ \Pr[\mathcal{A}(\eta, x) = 1 \mid x \leftarrow D^0_\eta] - \Pr[\mathcal{A}(\eta, x) = 1 \mid x \leftarrow D^1_\eta] \]

  is a negligible function of $\eta$.
- Denote $D^0 \approx D^1$. 
interface SingleEnv {
    bitstring getX();
}

interface SingleAdv {
    bit guess(SingleEnv envir);
}

class SingleInd\_{D_0, D_1} implements SingleEnv {
    private bitstring x;

    SingleInd\_{D_0, D_1}(bit b_0) {
        x ← D^{b_0};
    }

    bitstring getX() {
        return x;
    }
}

We have \((t, \varepsilon)\)-indistinguishability, if for all adversaries \(A\) that run in time \(t\) and implement SingleAdv,

\[
\left| \Pr[b \in_R \{0, 1\}; A.guess(new SingleInd_{D_0, D_1}(b)) = b] - \frac{1}{2} \right| \leq \varepsilon .
\]
With security parameter

interface SingleEnv {
    bitstring getX();
}

interface SingleAdv {
    bit guess(int η, SingleEnv envir);
}

class SingleInd_{D_0, D_1} implements SingleEnv {
    private bitstring x;
    SingleInd_{D_0, D_1}(int η, bit b_0) {
        x ← D_{η}^{b_0};
    }
    bitstring getX() {
        return x;
    }
}

We have (uniform polynomial) indistinguishability, if for all adversaries \( A \) that run in polynomial time (wrt. its first parameter) and implement SingleAdv,

\[
\left| \Pr[b \in_R \{0, 1\}; A.guess(new SingleInd_{D_0, D_1}(η, b)) = b] - \frac{1}{2} \right|
\]

is a negligible function of \( η \).
Transitivity

Theorem. If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

Code-based proof: We have to show that $\text{SingleInd}_{D^0,D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0,D^2}(\eta, 1)$. 
Transitivity

**Theorem.** If \( D^0 \approx D^1 \) and \( D^1 \approx D^2 \), then \( D^0 \approx D^2 \).

**Code-based proof:** We have to show that \( \text{SingleInd}_{D^0, D^2}(\eta, 0) \) may be replaced with \( \text{SingleInd}_{D^0, D^2}(\eta, 1) \).

```java
class SingleInd\(_{D^0, D^2}\) implements SingleEnv {
    private bitstring \( x \);
    bitstring \(\) \(\text{getX}()\) {
        return \( x \);
    }
    SingleInd\(_{D^0, D^2}(\text{int} \ \eta, \ \text{bit} \ b_0)\) {
        \( x \leftarrow D^2 \cdot b_0 \);
    }
} Call new SingleInd\(_{D^0, D^2}(\eta, 0)\)
Transitivity

**Theorem.** If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D^0, D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2}(\eta, 1)$.

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private bitstring x;

    SingleInd_{D^0, D^2}(\text{int} \ \eta, \ \text{bit} \ b_0) {
        x \leftarrow D^{2 \cdot b_0};
    }

    bitstring getX() {
        return x;
    }
}

Call \text{new} SingleInd_{D^0, D^2}(\eta, 0)
```

Propagate copies
Transitivity

Theorem. If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

Code-based proof: We have to show that $\text{SingleInd}_{D^0,D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0,D^2}(\eta, 1)$.

class SingleInd_{D^0,D^2} implements SingleEnv {
    private bitstring x;

    SingleInd_{D^0,D^2}(\text{int} \ \eta, \ \text{bit} \ \ b_0) {
        x ← D^0_\eta;
    }
}

Call new SingleInd_{D^0,D^2}(\eta, 0)
Transitivity

**Theorem.** If $D_0 \approx D_1$ and $D_1 \approx D_2$, then $D_0 \approx D_2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D_0,D_2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D_0,D_2}(\eta, 1)$.

```java
class SingleInd\(_{D_0,D_2}\) implements SingleEnv {
    private bitstring \(x\);
    bitstring \(x\) \(\leftarrow\) \(D_0^\eta\),
    \(x \leftarrow D_0^\eta;\)
}
```

Call `new SingleInd\(_{D_0,D_2}(\eta, 0)`

Keep \(x\) inside $\text{SingleInd}_{D_0,D_1}(\eta, 0)$
Transitivity

Theorem. If \( D^0 \approx D^1 \) and \( D^1 \approx D^2 \), then \( D^0 \approx D^2 \).

Code-based proof: We have to show that \( \text{SingleInd}_{D^0,D^2}(\eta, 0) \) may be replaced with \( \text{SingleInd}_{D^0,D^2}(\eta, 1) \).

```java
class SingleInd\_D^0,D^2 implements SingleEnv {
    private SingleEnv e;

    SingleInd\_D^0,D^2(int \eta, bit b0) {
        e := new SingleInd\_D^0,D^1(\eta, 0);
    }
    bitstring getX() {
        return e.getX();
    }
}
```

Call \( \text{new SingleInd}_{D^0,D^2}(\eta, 0) \)
Theorem. If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

Code-based proof: We have to show that $\text{SingleInd}_{D^0, D^2} (\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2} (\eta, 1)$.

class $\text{SingleInd}_{D^0, D^2}$ implements $\text{SingleEnv}$ {
    private $\text{SingleEnv}$ $e$;
    
    $\text{SingleInd}_{D^0, D^2} (\text{int} \ \eta, \ \text{bit} \ b_0)$ {
        $e := \text{new} \ \text{SingleInd}_{D^0, D^1} (\eta, 0)$;
    }
}

Call new $\text{SingleInd}_{D^0, D^2} (\eta, 0)$

Use $D^0 \approx D^1$
Transitivity

**Theorem.** If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D^0, D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2}(\eta, 1)$.

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private SingleEnv e;
    bitstring getX() {
        return e.getX();
    }
    SingleInd_{D^0, D^2}(int \eta, bit b_0) {
        e := new SingleInd_{D^0, D^1}(\eta, 1);
    }
}
Call new SingleInd_{D^0, D^2}(\eta, 0)
```
Transitivity

**Theorem.** If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D^0,D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0,D^2}(\eta, 1)$.

```java
class SingleInd_{D^0,D^2} implements SingleEnv {
    private SingleEnv e;
    bitstring getX() {
        return e.getX();
    }
    SingleInd_{D^0,D^2}(\text{int } \eta, \text{ bit } b_0) {
        e := \text{new } SingleInd_{D^0,D^1}(\eta, 1);
    }
}
```

Call $\text{new } \text{SingleInd}_{D^0,D^2}(\eta, 0)$

Take $x$ out of $\text{SingleInd}_{D^0,D^1}(\eta, 1)$
Transitivity

**Theorem.** If \( D^0 \approx D^1 \) and \( D^1 \approx D^2 \), then \( D^0 \approx D^2 \).

**Code-based proof:** We have to show that \( \text{SingleInd}_{D^0, D^2}(\eta, 0) \) may be replaced with \( \text{SingleInd}_{D^0, D^2}(\eta, 1) \).

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private bitstring x;
    bitstring getX() {
        return x;
    }
    SingleInd_{D^0, D^2}(int \eta, bit b0) {
        x ← D^1_{\eta};
    }
}
Call new SingleInd_{D^0, D^2}(\eta, 0)
```
Transitivity

Theorem. If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

Code-based proof: We have to show that $\text{SingleInd}_{D^0, D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2}(\eta, 1)$.

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private bitstring x;

    SingleInd_{D^0, D^2}(int \eta, bit b_0) {
        x \leftarrow D^1_\eta;
    }

    bitstring getX() {
        return x;
    }
}
```

Call `new SingleInd_{D^0, D^2}(\eta, 0)`

Keep $x$ inside $\text{SingleInd}_{D^1, D^2}(\eta, 0)$
Transitivity

Theorem. If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

Code-based proof: We have to show that $\text{SingleInd}_{D^0,D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0,D^2}(\eta, 1)$.

class SingleInd_{D^0,D^2} implements SingleEnv {
    private SingleEnv e;
    bitstring getX() {
        return e.getX();
    }
    SingleInd_{D^0,D^2}(\text{int} \ \eta, \ \text{bit} \ b_0) {
        e := \text{new} \ \text{SingleInd}_{D^1,D^2}(\eta, 0);
    }  
}  

Call $\text{new} \ \text{SingleInd}_{D^0,D^2}(\eta, 0)$
Transitivity

**Theorem.** If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D^0,D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0,D^2}(\eta, 1)$.

```java
class SingleInd_{D^0,D^2} implements SingleEnv {
    private SingleEnv e;
    SingleInd_{D^0,D^2}(int \eta, bit b_0) {
        e := new SingleInd_{D^1,D^2}(\eta, 0);
    }
}
```

Call `new SingleInd_{D^0,D^2}(\eta, 0)`

Use $D^1 \approx D^2$
Transitivity

**Theorem.** If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D^0, D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2}(\eta, 1)$.

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private SingleEnv e;

    SingleInd_{D^0, D^2}(int \eta, bit b_0) {
        e := new SingleInd_{D^1, D^2}(\eta, 1);
    }

    bitstring getX() {
        return e.getX();
    }
}
```

Call `new SingleInd_{D^0, D^2}(\eta, 0)`
Transitivity

**Theorem.** If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

**Code-based proof:** We have to show that $\text{SingleInd}_{D^0, D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2}(\eta, 1)$.

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private SingleEnv e;
    bitstring getX() {
        return e.getX();
    }
    SingleInd_{D^0, D^2}(int \eta, bit b_0) {
        e := new SingleInd_{D^1, D^2}(\eta, 1);
    }
}
```

Call `new SingleInd_{D^0, D^2}(\eta, 0)`

Take $x$ out of $\text{SingleInd}_{D^1, D^2}(\eta, 1)$
Transitivity

Theorem. If $D^0 \approx D^1$ and $D^1 \approx D^2$, then $D^0 \approx D^2$.

Code-based proof: We have to show that $\text{SingleInd}_{D^0, D^2}(\eta, 0)$ may be replaced with $\text{SingleInd}_{D^0, D^2}(\eta, 1)$.

```java
class SingleInd_{D^0, D^2} implements SingleEnv {
    private bitstring x;
    bitstring getX() {
        return x;
    }
    SingleInd_{D^0, D^2}(int \eta, bit b_0) {
        x ← D^2_{\eta};
    }
}
```

Call `new SingleInd_{D^0, D^2}(\eta, 0)`

This is what you get calling `new SingleInd_{D^0, D^2}(\eta, 1)`
Suppose that $D^0 \not\approx D^2$.

Let $A$ be a polynomial-time adversary such that $A$ can distinguish $D^0$ and $D^2$ with non-negligible advantage.

For $i \in \{0, 1, 2\}$, let

$$p^i_\eta = \Pr[A(\eta, x) = 1 \mid x \leftarrow D^i_\eta]$$

There is a polynomial $q$, such that for infinitely many $\eta$,

$$|p^0_\eta - p^2_\eta| \geq q(\eta).$$

For any such $\eta$, either $|p^0_\eta - p^1_\eta| \geq q(\eta)/2$ or $|p^1_\eta - p^2_\eta| \geq q(\eta)/2$.

Either $|p^0_\eta - p^1_\eta| \geq q(\eta)/2$ holds for infinitely many $\eta$, or $|p^1_\eta - p^2_\eta| \geq q(\eta)/2$ holds for infinitely many $\eta$.

$A$ distinguishes either $D^0$ and $D^1$, or $D^1$ and $D^2$. □
Independent components

Let $D^0, D^1, E$ be families of probability distributions.
Define the probability distribution $F^i_\eta$ by

1. Let $x \leftarrow D^i_\eta$.
2. Let $y \leftarrow E_\eta$.
3. Output $(x, y)$.

$E$ is polynomial-time constructible if there is a polynomial-time algorithm $\mathcal{E}$, such that the output of $\mathcal{E}(\eta)$ is distributed identically to $E_\eta$.

Theorem. If $D^0 \approx D^1$ and $E$ is polynomial-time constructible, then $F^0 \approx F^1$. 
class SingleInd_{F^0, F^1} implements SingleEnv {
    private bitstring x, y;

    SingleInd_{F^0, F^1}(int \eta, bit b_0) {
        x ← D^b_0\eta;
        y ← E\eta;
    }

    bitstring getX() {
        return (x, y);
    }
}

Call new SingleInd_{F^0, F^1}(\eta, 0)
Proof via code modification

class SingleInd_{F^0,F^1} implements SingleEnv {
    private bitstring x, y;
    bitstring getX() {
        return (x, y);
    }

    SingleInd_{F^0,F^1}(int η, bit b_0) {
        x ← D^{b_0}_η;
        y ← E_η;
    }
}

Call new SingleInd_{F^0,F^1}(η, 0)

Propagate copies
class SingleInd_{F_0,F_1} implements SingleEnv {
    private bitstring x, y;

    SingleInd_{F_0,F_1}(int \eta, bit b_0) {
        x \leftarrow D^0_{\eta};
        y \leftarrow E_{\eta};
    }

    bitstring getX() {
        return (x, y);
    }
}

Call new SingleInd_{F_0,F_1}(\eta, 0)
Proof via code modification

class SingleInd_{F_0,F_1} implements SingleEnv {
    private bitstring x, y;
    bitstring getX() {
        return (x, y);
    }
    SingleInd_{F_0,F_1}(int \eta, bit b_0) {
        x ← D_\eta^0;
        y ← E_\eta^1;
    }
}

Call new SingleInd_{F_0,F_1}(\eta, 0)

Keep x inside SingleInd_{D_0,D_1}(\eta, 0)
Proof via code modification

```java
class SingleInd\textsubscript{\(F_0,F_1\)} implements SingleEnv {
   private SingleEnv e;
   private bitstring y;
   
   SingleInd\textsubscript{\(F_0,F_1\)}(int \(\eta\), bit \(b_0\)) {
      e := new SingleInd\textsubscript{\(D_0,D_1\)}(\(\eta\), 0);
      y ← E_{\eta};
   }
   
   bitstring \text{getX}() {
      return (e.getX(), y);
   }
   
   Call new SingleInd\textsubscript{\(F_0,F_1\)}(\(\eta\), 0)
```
class SingleInd\(_{F_0,F_1}\) implements SingleEnv {
    private SingleEnv \(e\);
    private bitstring \(y\);

    SingleInd\(_{F_0,F_1}(\text{int } \eta, \text{ bit } b_0)\) {
        e := new SingleInd\(_{D_0,D_1}(\eta, 0)\);
        y ← \(E_\eta\);
    }

    bitstring \(\text{getX}()\) {
        return (e.getX(), y);
    }
}

Call new SingleInd\(_{F_0,F_1}(\eta, 0)\)

Use \(D_0 \approx D_1\)
Proof via code modification

class SingleInd_{F^0,F^1} implements SingleEnv {
    private SingleEnv e;
    private bitstring y;

    SingleInd_{F^0,F^1}(int \eta, bit b_0) {
        e := new SingleInd_{D^0,D^1}(\eta, 1);
        y ← E_\eta;
    }

    bitstring getX() {
        return (e.getX(), y);
    }
}

Call new SingleInd_{F^0,F^1}(\eta, 0)
Proof via code modification

class SingleInd_{F_0,F_1} implements SingleEnv {
    private SingleEnv e;
    private bitstring y;

    SingleInd_{F_0,F_1}(int \eta, bit b_0) {
        e := new SingleInd_{D_0,D_1}(\eta, 1);
        y ← E_\eta;
    }

    bitstring getX() {
        return (e.getX(), y);
    }
}

Call new SingleInd_{F_0,F_1}(\eta, 0)

Take \(x\) out again
Proof via code modification

class \textit{SingleInd}_{F_0,F_1} \ implements \textit{SingleEnv} 
{
    \text{private bitstring} \ x, y; 
    \text{bitstring} \ \text{getX}() \ \{ 
        return \ (x, y); 
    \} 
\
    \text{SingleInd}_{F_0,F_1}(\text{int} \ \eta, \ \text{bit} \ b_0) \ \{ 
        x \leftarrow D^1_\eta; 
        y \leftarrow E_\eta; 
    \} 
}\}

Call \texttt{new} \ \text{SingleInd}_{F_0,F_1}(\eta, 0)

This is equal to \texttt{new} \ \text{SingleInd}_{F_0,F_1}(\eta, 1)
“classical” proof

- Suppose that $F^0 \not\approx F^1$.
- Let $\mathcal{A}$ be a polynomial-time adversary such that $\mathcal{A}$ can distinguish $D^0$ and $D^1$ with non-negligible advantage.
  - $\mathcal{A}$ implements SimpleAdv

Define the adversary $\mathcal{B}$ implementing SimpleAdv:

```java
private SimpleAdv $\mathcal{A}$;

$\mathcal{B}$(SimpleAdv $\mathcal{A}_0$) {
  $\mathcal{A} := \mathcal{A}_0$;
}

bit guess(int $\eta$, SimpleEnv $e$) {
  ?????
}
```

- In `guess`, we could call $\mathcal{A}.guess(e)$.
- But if $e$ is $\text{SimpleInd}_{D^0,D^1}$ then the result probably won’t make much sense.
class PairEnv implements SimpleEnv {
    private SimpleEnv e;;
    private bitstring y;;

    PairEnv(int η, SimpleEnv e0) {
        e := e0;
        y ← Eη;
    }

    bitstring getX() {
        return (e.getX(), y);
    }
}
The adversary $\mathcal{B}$

class $\mathcal{B}$ implements SimpleAdv {
    private SimpleAdv $\mathcal{A}$;

    $\mathcal{B}$(SimpleAdv $\mathcal{A}_0$) {
        $\mathcal{A} := \mathcal{A}_0$;
    }

    bit guess(int $\eta$, SimpleEnv $e$) {
        return $\mathcal{A}$.guess(new PairEnv($\eta$, $e$));
    }
}

And now we have to argue that $\mathcal{B}$’s advantage really is the same as $\mathcal{A}$’s.
Multiple sampling

- Let \( D^0 = \{ D^0_\eta \}_{\eta \in \mathbb{N}} \) and \( D^1 = \{ D^1_\eta \}_{\eta \in \mathbb{N}} \) be two families of probability distributions.
- Let \( p \) be a positive polynomial.
- Let \( \vec{D}^b_\eta \) be a probability distribution over tuples

\[
(x_1, x_2, \ldots, x_{p(\eta)}) \in (\{0, 1\}^*)^{p(\eta)}
\]

such that

- each \( x_i \) is distributed according to \( D^b_\eta \);
- each \( x_i \) is independent of all other \( x \)-s.
Multiple sampling

Let \( D^0 = \{ D_\eta^0 \}_{\eta \in \mathbb{N}} \) and \( D^1 = \{ D_\eta^1 \}_{\eta \in \mathbb{N}} \) be two families of probability distributions.

Let \( p \) be a positive polynomial.

Let \( \vec{D}_\eta^b \) be a probability distribution over tuples

\[
(x_1, x_2, \ldots, x_{p(\eta)}) \in (\{0, 1\}^*)^{p(\eta)}
\]

such that

- each \( x_i \) is distributed according to \( D_\eta^b \);
- each \( x_i \) is is independent of all other \( x \)-s.

To sample \( \vec{D}_\eta^b \), sample \( D_\eta^b \) \( p(\eta) \) times and construct the tuple of sampled values.
Theorem. If $\vec{D}^0 \approx \vec{D}^1$ then $D^0 \approx D^1$. 
Theorem. If \( \vec{D}^0 \approx \vec{D}^1 \) then \( D^0 \approx D^1 \).
If \( \bullet \bullet \bullet \approx \bullet \bullet \bullet \) then \( \bullet \approx \bullet \).

Contrapositive: if \( \bullet \not\approx \bullet \) then \( \bullet \bullet \bullet \not\approx \bullet \bullet \bullet \)
\[ \tilde{\mathcal{D}} \text{-s indistinguishable } \Rightarrow \mathcal{D} \text{-s indistinguishable} \]

**Theorem.** If \( \tilde{\mathcal{D}}^0 \approx \tilde{\mathcal{D}}^1 \) then \( \mathcal{D}^0 \approx \mathcal{D}^1 \).

If \( \bullet \bullet \bullet \approx \bullet \bullet \bullet \) then \( \bullet \approx \bullet \).

Contrapositive: if \( \bullet \not\approx \bullet \) then \( \bullet \bullet \bullet \not\approx \bullet \bullet \bullet \).

If \( \bullet \not\approx \bullet \) then there exists a PPT distinguisher \( \mathcal{A} \):

\[
\Pr[b = b^* \mid b \in_R \{0, 1\}, x \leftarrow D^b_\eta, b^* \leftarrow \mathcal{A}(\eta, x)] \geq 1/2 + 1/q(\eta)
\]

for some polynomial \( q \) and infinitely many \( \eta \).
Theorem. If $\vec{D}^0 \approx \vec{D}^1$ then $D^0 \approx D^1$.
If $\bullet\bullet\bullet \approx \bullet\bullet\bullet$ then $\bullet \approx \bullet$.

Contrapositive: if $\bullet \not\approx \bullet$ then $\bullet\bullet\bullet \not\approx \bullet\bullet\bullet$
If $\bullet \not\approx \bullet$ then there exists a PPT distinguisher $\mathcal{A}$:

$$\Pr[\mathcal{A}(\eta, x) = 0 \mid x \leftarrow D^0_\eta] - \Pr[\mathcal{A}(\eta, x) = 0 \mid x \leftarrow D^1_\eta] \geq 2/q(\eta)$$

for some polynomial $q$ and infinitely many $\eta$. 
Theorem. If $\vec{D}^0 \approx \vec{D}^1$ then $D^0 \approx D^1$.
If $\bullet \bullet \approx \bullet \bullet \bullet$ then $\bullet \approx \bullet$.

Contrapositive: if $\bullet \not\approx \bullet$ then $\bullet \bullet \bullet \not\approx \bullet \bullet \bullet$
If $\bullet \not\approx \bullet$ then there exists a PPT distinguisher $A$:

$$\Pr[A(\eta, x) = 0 \mid x \leftarrow D^0_\eta] - \Pr[A(\eta, x) = 0 \mid x \leftarrow D^1_\eta] \geq 1/q(\eta)$$

for some polynomial $q$ and infinitely many $\eta$. 

$\vec{D}$-s indistinguishable $\Rightarrow$ $D$-s indistinguishable
If $\tilde{D}^0 \approx \tilde{D}^1$ then $D^0 \approx D^1$.
If $\bullet\bullet\bullet \approx \bullet\bullet\bullet$ then $\bullet \approx \bullet$.

Contrapositive: if $\bullet \not\approx \bullet$ then $\bullet\bullet\bullet \not\approx \bullet\bullet\bullet$
If $\bullet \not\approx \bullet$ then there exists a PPT distinguisher $\mathcal{A}$:

$$\Pr[\mathcal{A}(\eta, x) = 0 \mid x \leftarrow D^0_\eta] - \Pr[\mathcal{A}(\eta, x) = 0 \mid x \leftarrow D^1_\eta] \geq 1/q(\eta)$$

for some polynomial $q$ and infinitely many $\eta$.

Let $\mathcal{B}(\eta, (x_1, \ldots, x_{p(\eta)})) = \mathcal{A}(\eta, x_1)$.
Then $\mathcal{B}$ distinguishes $\bullet\bullet\bullet$ and $\bullet\bullet\bullet$. 
\[ \tilde{D} - \text{s indistinguishable} \Rightarrow D - \text{s indistinguishable} \]

**Theorem.** If \( \tilde{D}^0 \approx \tilde{D}^1 \) then \( D^0 \approx D^1 \).
If \( \bullet \bullet \approx \bullet \bullet \bullet \) then \( \bullet \approx \bullet \).

**Contrapositive:** if \( \bullet \not\approx \bullet \) then \( \bullet \bullet \not\approx \bullet \bullet \bullet \)
If \( \bullet \not\approx \bullet \) then there exists a PPT distinguisher \( \mathcal{A} \):

\[
\Pr[\mathcal{A}(\eta, x) = 0 \mid x \leftarrow D^0_\eta] - \Pr[\mathcal{A}(\eta, x) = 0 \mid x \leftarrow D^1_\eta] \geq 1/q(\eta)
\]

for some polynomial \( q \) and infinitely many \( \eta \).

Let \( \mathcal{B}(\eta, (x_1, \ldots, x_{p(\eta)})) = \mathcal{A}(\eta, x_1) \).
Then \( \mathcal{B} \) distinguishes \( \bullet \bullet \bullet \) and \( \bullet \bullet \bullet \bullet \).

I.e. we can distinguish \( \bullet \bullet \bullet \) from \( \bullet \bullet \bullet \bullet \) by just considering the first elements of the tuples.
\(D\)-s indistinguishable \(\Rightarrow \vec{D}\)-s indistinguishable

(Interesting) theorem. If \(D^0 \approx D^1\) and there exist polynomial-time algorithms \(\mathcal{D}^0\) and \(\mathcal{D}^1\), such that the output distribution of \(\mathcal{D}^b(\eta)\) is equal to \(D^b_\eta\), then \(\vec{D}^0 \approx \vec{D}^1\).
(Interesting) theorem. If $D^0 \approx D^1$ and there exist polynomial-time algorithms $D^0$ and $D^1$, such that the output distribution of $D^b(\eta)$ is equal to $D^b_{\eta}$, then $\vec{D}^0 \approx \vec{D}^1$.

Assume for now that the polynomial $p$ is a constant. I.e. the length of the vector $\vec{x}$ does not depend on the security parameter $\eta$.

Let $p$ be the common value of $p(\eta)$ for all $\eta$.

Theorem statement: if $\bullet \approx \bullet$ then $\bullet\bullet\bullet \approx \bullet\bullet\bullet$. (let $p = 3$)
**D-s indistinguishable ⇒ \( \vec{D} \)-s indistinguishable**

(Interesting) theorem. If \( D^0 \approx D^1 \) and there exist polynomial-time algorithms \( D^0 \) and \( D^1 \), such that the output distribution of \( D^b(\eta) \) is equal to \( D^b_{\eta} \), then \( \vec{D}^0 \approx \vec{D}^1 \).

Assume for now that the polynomial \( p \) is a constant. I.e. the length of the vector \( \vec{x} \) does not depend on the security parameter \( \eta \). Let \( p \) be the common value of \( p(\eta) \) for all \( \eta \).

Theorem statement: if \( \bullet \approx \bullet \) then \( \bullet \bullet \bullet \approx \bullet \bullet \bullet \). (let \( p = 3 \))

Our lemmas said \( (\bullet \approx \bullet \land \bullet \approx \bullet) \Rightarrow \bullet \approx \bullet \) and \( \bullet \approx \bullet \Rightarrow \bullet \bullet \approx \bullet \bullet \).
(Interesting) theorem. If $D^0 \approx D^1$ and there exist polynomial-time algorithms $D^0$ and $D^1$, such that the output distribution of $D^b(\eta)$ is equal to $D^b_\eta$, then $\vec{D}^0 \approx \vec{D}^1$.

Assume for now that the polynomial $p$ is a constant. I.e. the length of the vector $\vec{x}$ does not depend on the security parameter $\eta$. Let $p$ be the common value of $p(\eta)$ for all $\eta$.

Theorem statement: if $\bullet \approx \bullet$ then $\bullet\bullet\bullet \approx \bullet\bullet\bullet$. (let $p = 3$)

Our lemmas said $(\bullet \approx \bullet \wedge \bullet \approx \bullet) \Rightarrow \bullet \approx \bullet$ and $\bullet \approx \bullet \Rightarrow \bullet\bullet \approx \bullet\bullet$.
\( \mathcal{D} \)-s indistinguishable \( \Rightarrow \tilde{\mathcal{D}} \)-s indistinguishable

(Interesting) theorem. If \( \mathcal{D}^0 \approx \mathcal{D}^1 \) and there exist polynomial-time algorithms \( \mathcal{D}^0 \) and \( \mathcal{D}^1 \), such that the output distribution of \( \mathcal{D}^b(\eta) \) is equal to \( \mathcal{D}^b_\eta \), then \( \tilde{\mathcal{D}}^0 \approx \tilde{\mathcal{D}}^1 \).

Assume for now that the polynomial \( p \) is a constant. I.e. the length of the vector \( \vec{x} \) does not depend on the security parameter \( \eta \).
Let \( p \) be the common value of \( p(\eta) \) for all \( \eta \).

Theorem statement: if \( \bullet \approx \bullet \) then \( \bullet \bullet \bullet \approx \bullet \bullet \bullet \). (let \( p = 3 \))

Our lemmas said \( (\bullet \approx \bullet \land \bullet \approx \bullet) \Rightarrow \bullet \approx \bullet \) and \( \bullet \approx \bullet \Rightarrow \bullet \bullet \approx \bullet \bullet \).

\( \bullet \bullet \bullet \approx \bullet \bullet \bullet \)
$D$-s indistinguishable $\Rightarrow \vec{D}$-s indistinguishable

(Interesting) theorem. If $D^0 \approx D^1$ and there exist polynomial-time algorithms $D^0$ and $D^1$, such that the output distribution of $D^b_\eta$ is equal to $D^b_\eta$, then $\vec{D}^0 \approx \vec{D}^1$.

Assume for now that the polynomial $p$ is a constant. I.e. the length of the vector $\vec{x}$ does not depend on the security parameter $\eta$.
Let $p$ be the common value of $p(\eta)$ for all $\eta$.

Theorem statement: if $\bullet \approx \bullet$ then $\bullet\bullet\bullet \approx \bullet\bullet\bullet$. (let $p = 3$)

Our lemmas said $(\bullet \approx \bullet \land \bullet \approx \bullet) \Rightarrow \bullet \approx \bullet$ and $\bullet \approx \bullet \Rightarrow \bullet\bullet \approx \bullet\bullet$.

$\bullet\bullet\bullet \approx \bullet\bullet\bullet \approx \bullet\bullet\bullet$
$D$-s indistinguishable $\Rightarrow \vec{D}$-s indistinguishable

(Interesting) theorem. If $D^0 \approx D^1$ and there exist polynomial-time algorithms $D^0$ and $D^1$, such that the output distribution of $D^b(\eta)$ is equal to $D^b_\eta$, then $\vec{D}^0 \approx \vec{D}^1$.

Assume for now that the polynomial $p$ is a constant. I.e. the length of the vector $\vec{x}$ does not depend on the security parameter $\eta$.
Let $p$ be the common value of $p(\eta)$ for all $\eta$.

Theorem statement: if $\bullet \approx \bullet$ then $\bullet\bullet \approx \bullet\bullet$. (let $p = 3$)

Our lemmas said $(\bullet \approx \bullet \wedge \bullet \approx \bullet) \Rightarrow \bullet \approx \bullet$ and $\bullet \approx \bullet \Rightarrow \bullet\bullet \approx \bullet\bullet$.

$\bullet\bullet \bullet \approx \bullet\bullet\bullet \approx \bullet\bullet\bullet \approx \bullet\bullet\bullet$. 
\(D\)-s indistinguishable \(\Rightarrow \vec{D}\)-s indistinguishable

(Interesting) theorem. If \(D^0 \approx D^1\) and there exist polynomial-time algorithms \(\mathcal{D}^0\) and \(\mathcal{D}^1\), such that the output distribution of \(\mathcal{D}^b(\eta)\) is equal to \(D^b_\eta\), then \(\vec{D}^0 \approx \vec{D}^1\).

Assume for now that the polynomial \(p\) is a constant. I.e. the length of the vector \(\vec{x}\) does not depend on the security parameter \(\eta\).

Let \(p\) be the common value of \(p(\eta)\) for all \(\eta\).

Theorem statement: if \(\bullet \approx \bullet\) then \(\bullet\bullet\bullet \approx \bullet\bullet\bullet\). (let \(p = 3\))

Our lemmas said \((\bullet \approx \bullet \land \bullet \approx \bullet) \Rightarrow \bullet \approx \bullet\) and \(\bullet \approx \bullet \Rightarrow \bullet\bullet \approx \bullet\bullet\).

\(\bullet\bullet\bullet \approx \bullet\bullet\bullet \approx \bullet\bullet\bullet \approx \bullet\bullet\bullet\). By transitivity, \(\bullet\bullet\bullet \approx \bullet\bullet\bullet\).

(Actually, we’re done with this case)
Constructing the distinguisher

Contrapositive: if $\bullet\bullet\neq \bullet\bullet$ then $\bullet \neq \bullet$. 
Constructing the distinguisher

Contrapositive: if \( \bullet \bullet \not\approx \bullet \bullet \) then \( \bullet \not\approx \bullet \).

If \( \bullet \bullet \not\approx \bullet \bullet \) then there exists a PPT distinguisher \( A \):

\[
\Pr[ \mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{D}_0^{\eta}] - \Pr[ \mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{D}_1^{\eta}] \geq 1/q(\eta)
\]

for some polynomial \( q \) and infinitely many \( \eta \).
Hybrid distributions

If $\bullet \bullet \not\approx \bullet \bullet \bullet$ then

$$(\bullet \bullet \not\approx \bullet \bullet \bullet) \lor (\bullet \bullet \bullet \not\approx \bullet \bullet \bullet) \lor (\bullet \bullet \bullet \not\approx \bullet \bullet \bullet)$$
Hybrid distributions

If \( \bullet \bullet \not\approx \bullet \bullet \) then

\[
(\bullet \bullet \not\approx \bullet \bullet) \lor (\bullet \bullet \not\approx \bullet \bullet) \lor (\bullet \bullet \not\approx \bullet \bullet)
\]

Let \( \vec{E}^k_\eta \), where \( 0 \leq k \leq p \), be a probability distribution over tuples \((x_1, \ldots, x_p)\), where

- each \( x_i \) is independent of all other \( x \)-s;
- \( x_1, \ldots, x_k \) are distributed according to \( D^0_\eta \);
- \( x_{k+1}, \ldots, x_p \) are distributed according to \( D^1_\eta \).

Thus \( \vec{E}^0_\eta = \vec{D}^1_\eta \) and \( \vec{E}^p_\eta = \vec{D}^0_\eta \). Define \( P^k_\eta = \Pr[\mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{E}^k_\eta] \). Then for infinitely many \( \eta \):

\[
\frac{1}{q(\eta)} \leq P^p_\eta - P^0_\eta = \sum_{i=1}^{p} (P^i_\eta - P^{i-1}_\eta) .
\]

And for some \( j_\eta \), \( P^{j_\eta}_\eta - P^{j_\eta-1}_\eta \geq \frac{1}{(p \cdot q(\eta))} \).
\( A \) distinguishes hybrids

There exists \( j \), such that \( j = j_\eta \) for infinitely many \( \eta \). Thus

\[
\Pr[\mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{E}_\eta^j] - \Pr[\mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{E}_\eta^{j-1}] \geq 1/(p \cdot q(\eta))
\]

for infinitely many \( \eta \). We have \( \vec{E}^{j-1} \not\approx \vec{E}^j \).
There exists \( j \), such that \( j = j_\eta \) for infinitely many \( \eta \). Thus

\[
\Pr[\mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{E}_\eta^j] - \Pr[\mathcal{A}(\eta, \vec{x}) = 0 \mid \vec{x} \leftarrow \vec{E}_\eta^{j-1}] \geq 1/(p \cdot q(\eta))
\]

for infinitely many \( \eta \). We have \( \vec{E}_\eta^{j-1} \not\approx \vec{E}_\eta^j \).

If we can distinguish

\[
\vec{E}_\eta^j = \begin{array}{c}
\bullet \cdots \bullet
\end{array}
\]

from

\[
\vec{E}_\eta^{j-1} = \begin{array}{c}
\bullet \cdots \bullet
\end{array}
\]

using \( \mathcal{A} \), then how do we distinguish \( \bullet \) and \( \bullet \)?
Distinguisher for $D^0$ and $D^1$

On input $(\eta, x)$:

1. Let $x_1 := D^0(\eta), \ldots, x_{j-1} := D^0(\eta)$.
2. Let $x_j := x$
3. Let $x_{j+1} := D^1(\eta), \ldots, x_p := D^1(\eta)$
4. Let $\vec{x} = (x_1, \ldots, x_p)$.
5. Call $b^* := A(\eta, \vec{x})$ and return $b^*$.

The advantage of this distinguisher is at least $1/(p \cdot q(\eta))$. 
Distinguisher for $D^0$ and $D^1$

On input $(\eta, x)$:

1. Let $x_1 := D^0(\eta), \ldots, x_{j-1} := D^0(\eta)$.
2. Let $x_j := x$
3. Let $x_{j+1} := D^1(\eta), \ldots, x_p := D^1(\eta)$
4. Let $\bar{x} = (x_1, \ldots, x_p)$.
5. Call $b^* := A(\eta, \bar{x})$ and return $b^*$.

The advantage of this distinguisher is at least $1/(p \cdot q(\eta))$.

Unfortunately, the above construction was not constructive.
For infinitely many $\eta$ we had

$$1/q(\eta) \leq P^p_\eta - P^0_\eta = \sum_{i=1}^{p} (P^i_\eta - P^{i-1}_\eta).$$

Hence the average value of $P^j_\eta - P^{j-1}_\eta$ is $\geq 1/(p \cdot q(\eta))$. 
Being constructive

For infinitely many $\eta$ we had

$$
1/q(\eta) \leq P^p_\eta - P^0_\eta = \sum_{i=1}^{p} (P^i_\eta - P^{i-1}_\eta).
$$

Hence the average value of $P^j_\eta - P^{j-1}_\eta$ is $\geq 1/(p \cdot q(\eta))$.

Consider the following distinguisher $B(\eta, x)$:

1. Let $j \in R \{1, \ldots, p\}$.
2. Let $x_1 := D^0(\eta), \ldots, x_{j-1} := D^0(\eta)$.
3. Let $x_j := x$
4. Let $x_{j+1} := D^1(\eta), \ldots, x_p := D^1(\eta)$
5. Let $\bar{x} = (x_1, \ldots, x_p)$.
6. Call $b^* := A(\eta, \bar{x})$ and return $b^*$. 
What $\mathcal{B}$ does

If (for example) $p = 5$, then $\mathcal{B}$ tries to distinguish

- $\bullet\bullet\bullet\bullet\bullet$ and $\bullet\bullet\bullet\bullet\bullet$ with probability $1/5$
- $\bullet\bullet\bullet$ and $\bullet\bullet\bullet\bullet\bullet$ with probability $1/5$
- $\bullet\bullet\bullet\bullet$ and $\bullet\bullet\bullet\bullet\bullet$ with probability $1/5$
- $\bullet\bullet\bullet\bullet\bullet$ and $\bullet\bullet\bullet\bullet\bullet$ with probability $1/5$
- $\bullet\bullet\bullet\bullet\bullet$ and $\bullet\bullet\bullet\bullet\bullet$ with probability $1/5$
- $\bullet\bullet\bullet\bullet\bullet$ and $\bullet\bullet\bullet\bullet\bullet$ with probability $1/5$

The advantage of $\mathcal{B}$ is $1/p$ times the sum of $\mathcal{A}$’s advantages of distinguishing these pairs of distributions.

The advantage of $\mathcal{B}$ is

$$
\frac{1}{p} \sum_{j=1}^{p} P_{\eta}^j - P_{\eta}^{j-1} = \frac{1}{p} (P_{\eta}^p - P_{\eta}^0) \geq \frac{1}{p \cdot q(\eta)} .
$$
If $p$ depends on $\eta$

$\mathcal{B}(\eta, x)$ is:

1. Let $j \in_R \{1, \ldots, p(\eta)\}$.
2. Let $x_1 := D^0(\eta), \ldots, x_{j-1} := D^0(\eta)$.
3. Let $x_j := x$
4. Let $x_{j+1} := D^1(\eta), \ldots, x_{p(\eta)} := D^1(\eta)$
5. Let $\vec{x} = (x_1, \ldots, x_{p(\eta)})$.
6. Call $b^* := A(\eta, \vec{x})$ and return $b^*$.

The advantage of $\mathcal{B}$ is at least $1/(p(\eta) \cdot q(\eta))$. 