CPA-secure encryption from a PRF:

$$\Sigma[F]:$$

$$\mathcal{K} = \{0,1\}^{\lambda} \qquad \frac{\operatorname{Enc}(k,m):}{r \leftarrow \{0,1\}^{\lambda}}$$

$$\mathcal{C} = \{0,1\}^{\lambda} \times \{0,1\}^{\operatorname{out}} \qquad x := F(k,r) \oplus m$$

$$\operatorname{return}(r,x)$$

$$\frac{\operatorname{KeyGen:}}{k \leftarrow \{0,1\}^{\lambda}}$$

$$\operatorname{return} k \qquad \frac{\operatorname{Dec}(k,(r,x)):}{m := F(k,r) \oplus x}$$

$$\operatorname{return} m$$

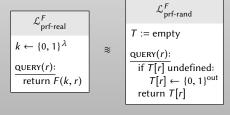
Claim:

If F is a secure PRF (with in = λ) then Σ is a CPA\$-secure encryption scheme. That is, $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$.

Overview:

Want to show:

The proof will **use** the fact *F* is a secure PRF. In other words,



$$\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$$

$$k \leftarrow \{0, 1\}^{\lambda}$$

CHALLENGE(m):

$$r \leftarrow \{0,1\}^{\lambda}$$
$$x := F(k,r) \oplus m$$
$$return (r,x)$$

Starting point is $\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$.



$$\mathcal{L}_{\mathsf{cpa\$-real}}^{\Sigma}$$

$$k \leftarrow \{0,1\}^{\lambda}$$

$$k \leftarrow \{0,1\}^{\lambda}$$

$$r \leftarrow \{0,1\}^{\lambda}$$

 $x := F(k, r) \oplus m$

return (r, x)

Starting point is $\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$. Factor out call to F.

```
CHALLENGE(m):
r \leftarrow \{0,1\}^{\lambda}
z := QUERY(r) \diamond \qquad k \leftarrow \{0,1\}^{\lambda}
CHALLENGE(m):
                                QUERY(r):
   x := z \oplus m
                                    return F(k,r)
   return (r, x)
```

Starting point is $\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$. Factor out call to F.



CHALLENGE(m):

$$r \leftarrow \{0,1\}^{\lambda}$$

 $z := \text{QUERY}(r)$
 $x := z \oplus m$
return (r,x)

$$\mathcal{L}_{prf-real}^{F}$$

 $k \leftarrow \{0,1\}^{\lambda}$
QUERY (r) :
return $F(k, r)$

$$\begin{array}{c}
\mathcal{L}_{prf-real}^{F} \\
k \leftarrow \{0,1\}^{\lambda} \\
\underline{\text{QUERY}(r):} \\
\underline{\text{return } F(k,r)}
\end{array}$$

Starting point is $\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$. Factor out call to F.

```
CHALLENGE(m):
  r \leftarrow \{0,1\}^{\lambda}
  z := QUERY(r) \diamond
  x := z \oplus m
  return (r, x)
```

```
\mathcal{L}^F_{\text{prf-rand}}
T := empty
QUERY(r):
   if T[r] undefined:
     T[r] \leftarrow \{0,1\}^{\text{out}}
   return T[r]
```

Apply security of F: replace $\mathcal{L}_{prf-real}$ with $\mathcal{L}_{prf-rand}$.



CHALLENGE(m): $r \leftarrow \{0,1\}^{\lambda}$ $z := QUERY(r) |\diamond|$ $x := z \oplus m$ return (r, x)

 $\mathcal{L}^F_{ ext{prf-rand}}$ T := emptyQUERY(r): if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

Apply security of F: replace $\mathcal{L}_{prf-real}$ with $\mathcal{L}_{prf-rand}$. Are we done?



CHALLENGE(m): $r \leftarrow \{0,1\}^{\lambda}$ $z := QUERY(r) \diamond$ $x := z \oplus m$ return (r, x)

```
\mathcal{L}_{prf-rand}^{F}
T := empty
QUERY(r):
   if T[r] undefined:

T[r] \leftarrow \{0, 1\}^{\text{out}}
   return T[r]
```

If *r* happens to repeat (which is possible), one-time pad *z* is reused!



CHALLENGE(m): $r \leftarrow \{0,1\}^{\lambda}$ $z := QUERY(r) |\diamond|$ $x := z \oplus m$ return (r, x)

$$\mathcal{L}_{prf-rand}^{F}$$

$$T := empty$$

$$\frac{QUERY(r):}{\text{if } T[r] \text{ undefined:}}$$

$$T[r] \leftarrow \{0,1\}^{\text{out}}$$

$$\text{return } T[r]$$

Must use fact that *r* is unlikely to repeat (when chosen this way)



CHALLENGE(m): r := SAMP() $z := QUERY(r) |\diamond|$ $x := z \oplus m$ return (r, x)

 $\mathcal{L}^F_{\text{prf-rand}}$ T := emptyQUERY(r): if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

 $\mathcal{L}_{\mathsf{samp-L}}$ $r \leftarrow \{0,1\}^{\lambda}$ return r

Isolate sampling of r.



r := SAMP() $z := QUERY(r) |\diamond|$

 $x := z \oplus m$

return (r, x)

 $\mathcal{L}^F_{\text{prf-rand}}$

T := empty

QUERY(r):

if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

 $\mathcal{L}_{\mathsf{samp-L}}$

SAMP():

 $r \leftarrow \{0,1\}^{\lambda}$ return r

Isolate sampling of r.



CHALLENGE(m):

r := SAMP()

 $z := QUERY(r) |\diamond|$

 $x := z \oplus m$

return (r, x)

 $\mathcal{L}_{prf-rand}^{F}$

T := empty

QUERY(r):

if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

 $\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

SAMP():

 $r \leftarrow \{0,1\}^{\lambda} \setminus R$ $R := R \cup \{r\}$

return r

Sample *r* without replacement (change \mathcal{L}_{samp-L} to \mathcal{L}_{samp-R}).



CHALLENGE(m):

r := SAMP() $z := QUERY(r) \Leftrightarrow$ $x := z \oplus m$ return (r, x)

$\mathcal{L}_{\mathsf{prf-rand}}^{F}$

T := empty

QUERY(r):

if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

$\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

SAMP():

$$r \leftarrow \{0,1\}^{\lambda} \setminus R$$
$$R := R \cup \{r\}$$
$$return r$$

Sample r without replacement (change \mathcal{L}_{samp-L} to \mathcal{L}_{samp-R}).



r := SAMP()

 $z := QUERY(r) |\diamond|$ $x := z \oplus m$

return (r, x)

$\mathcal{L}_{prf-rand}^{F}$

T := empty

QUERY(r):

if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

$\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

SAMP():

$$r \leftarrow \{0,1\}^{\lambda} \setminus R$$
$$R := R \cup \{r\}$$
$$return r$$

Now *r* values are **guaranteed** to never repeat.



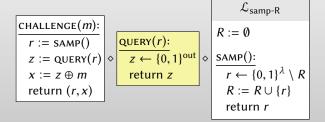
CHALLENGE(m): r := SAMP() $z := QUERY(r) |\diamond|$ $x := z \oplus m$ return (r, x)

 $\mathcal{L}_{prf-rand}^{F}$ T := emptyQUERY(r): if T[r] undefined: $T[r] \leftarrow \{0,1\}^{\text{out}}$ return T[r]

 $\mathcal{L}_{\mathsf{samp-R}}$ $R := \emptyset$ SAMP(): $r \leftarrow \{0,1\}^{\lambda} \setminus R$ $R := R \cup \{r\}$ return r

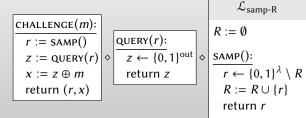
If-statement is always taken.





Middle library can therefore be simplified.





Middle library can therefore be simplified.



r := SAMP() $z := QUERY(r) |\diamond|$

 $x := z \oplus m$

return (r, x)

QUERY(r): $z \leftarrow \overline{\{0,1\}}^{\text{out}} \diamond$ return z

 $\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

SAMP():

 $r \leftarrow \{0,1\}^{\lambda} \setminus R$

 $R:=R\cup\{r\}$

return r

Inline call to QUERY.



r := SAMP() $z \leftarrow \{0,1\}^{\text{out}} \diamond$ $x := z \oplus m$ return (r, x)

$\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

SAMP():

$$\begin{array}{c}
 \hline
 r \leftarrow \{0,1\}^{\lambda} \setminus R \\
 R := R \cup \{r\} \\
 \text{return } r
\end{array}$$

Inline call to QUERY.



r := SAMP() $z \leftarrow \{0,1\}^{\text{out}} \diamond$ $x := z \oplus m$ return (r, x)

$\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

SAMP():

$$\begin{array}{c}
 \hline
 r \leftarrow \{0,1\}^{\lambda} \setminus R \\
 R := R \cup \{r\} \\
 return r
\end{array}$$

Inline call to QUERY.



CHALLENGE(m): r := SAMP() $z \leftarrow \{0,1\}^{\text{out}} \Leftrightarrow \text{SAMP}()$: return (r, x)

$$\frac{\text{HALLENGE}(m):}{r := \text{SAMP}()} \\
z \leftarrow \{0, 1\}^{\text{out}} \\
x := z \oplus m \\
\text{return } (r, x)$$

$$\downarrow \frac{\text{SAMP}():}{r \leftarrow \{0, 1\}^{\lambda} \setminus R} \\
R := R \cup \{r\} \\
\text{return } r$$

Can apply the "one-time pad rule" (since mask z is uniform each time)



 $\mathcal{L}_{\mathsf{samp-R}}$ CHALLENGE(m): $R := \emptyset$ r := SAMP() $x \leftarrow \{0, 1\}^{out} \Leftrightarrow \frac{SAMP():}{r \leftarrow 1}$ $r \leftarrow \{0,1\}^{\lambda} \setminus R$ return (r, x) $R := R \cup \{r\}$ return r

Can apply the "one-time pad rule" (since mask z is uniform each time)



r := SAMP() $x \leftarrow \{0, 1\}^{out} \diamond \frac{SAMP():}{\pi}$ return (r, x)

 $\mathcal{L}_{\mathsf{samp-R}}$

 $R := \emptyset$

 $r \leftarrow \{0,1\}^{\lambda} \setminus R$ $R := R \cup \{r\}$

return r

Can apply the "one-time pad rule" (since mask z is uniform each time)



```
CHALLENGE(m):
  r := SAMP()
 x \leftarrow \{0,1\}^{\text{out}} \diamond
  return (r, x)
```

$$\mathcal{L}_{samp-L}$$

$$\frac{samp():}{r \leftarrow \{0,1\}^{\lambda}}$$

$$return r$$

Replace \mathcal{L}_{samp-L} with \mathcal{L}_{samp-R} .



CHALLENGE(m): r := SAMP() $x \leftarrow \{0,1\}^{\text{out}} | \diamond |$ return (r, x)

 $\mathcal{L}_{\mathsf{samp-L}}$ SAMP(): $r \leftarrow \{0,1\}^{\lambda}$ return r

Replace \mathcal{L}_{samp-L} with \mathcal{L}_{samp-R} .



 $\mathcal{L}_{\mathsf{samp-L}}$ CHALLENGE(m): r := SAMP() $x \leftarrow \{0, 1\}^{out}$ \diamond SAMP(): $r \leftarrow \{0,1\}^{\lambda}$ return (r, x)return r

Inline call to SAMP.



CHALLENGE(m): $r \leftarrow \{0,1\}^{\lambda}$ $x \leftarrow \{0,1\}^{\text{out}}$ return (r, x)

Inline call to SAMP.



CHALLENGE(m): $r \leftarrow \{0, 1\}^{\lambda}$ $x \leftarrow \{0, 1\}^{\text{out}}$ return (r, x)

Inline call to SAMP.



CHALLENGE(m): $r \leftarrow \{0,1\}^{\lambda}$

 $x \leftarrow \{0, 1\}^{\text{out}}$ return (r, x)

But every response is chosen uniformly: This is just $\mathcal{L}_{cpa\$-rand}$.