

# CPA-secure encryption from a PRF:

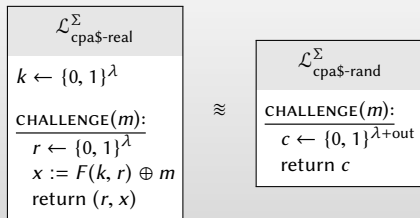
$\Sigma[F]$ :	
$\mathcal{K} = \{0, 1\}^\lambda$	<u>Enc(<math>k, m</math>):</u>
$\mathcal{M} = \{0, 1\}^{\text{out}}$	$r \leftarrow \{0, 1\}^\lambda$
$\mathcal{C} = \{0, 1\}^\lambda \times \{0, 1\}^{\text{out}}$	$x := F(k, r) \oplus m$
	return $(r, x)$
<u>KeyGen:</u>	<u>Dec(<math>k, (r, x)</math>):</u>
$k \leftarrow \{0, 1\}^\lambda$	$m := F(k, r) \oplus x$
return $k$	return $m$

## Claim:

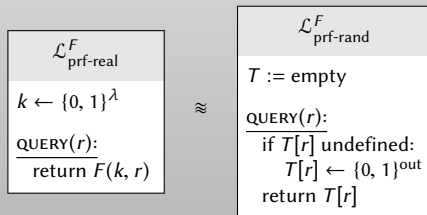
If  $F$  is a secure PRF (with  $\text{in} = \lambda$ ) then  $\Sigma$  is a CPA-secure encryption scheme. That is,  $\mathcal{L}_{\text{cpa-real}}^\Sigma \approx \mathcal{L}_{\text{cpa-rand}}^\Sigma$ .

# Overview:

Want to show:



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



# Security proof



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

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

CHALLENGE( $m$ ):

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

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

return ( $r, x$ )

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

# Security proof

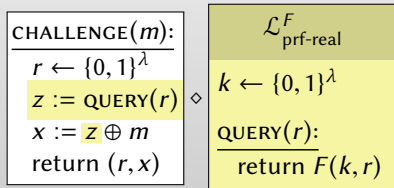

$$\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$$
$$k \leftarrow \{0, 1\}^{\lambda}$$

CHALLENGE( $m$ ):

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

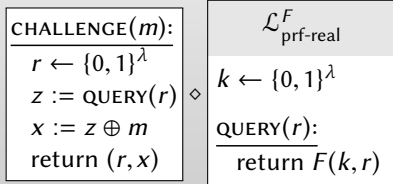
Starting point is  $\mathcal{L}_{\text{cpa\$-real}}^{\Sigma}$ . Factor out call to  $F$ .

# Security proof



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

# Security proof



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

# Security proof



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



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

$T := \text{empty}$

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

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

# Security proof



<b>CHALLENGE(<math>m</math>):</b>
$r \leftarrow \{0, 1\}^\lambda$
$z := \text{QUERY}(r)$
$x := z \oplus m$
return $(r, x)$



$\mathcal{L}_{\text{prf-rand}}^F$
$T := \text{empty}$
<b>QUERY(<math>r</math>):</b>
if $T[r]$ undefined:
$T[r] \leftarrow \{0, 1\}^{\text{out}}$
return $T[r]$

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



# Security proof



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



$\mathcal{L}_{\text{prf-rand}}^F$   
 $T := \text{empty}$   
**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!

# Security proof



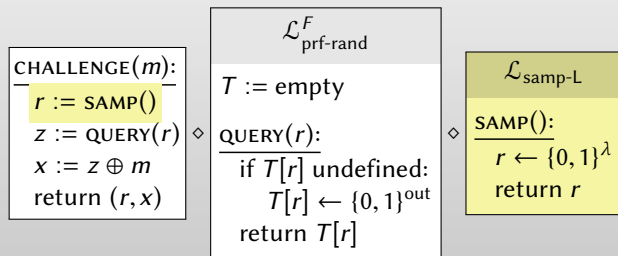
**CHALLENGE( $m$ ):**  
 $r \leftarrow \{0, 1\}^\lambda$   
 $z := \text{QUERY}(r)$   
 $x := z \oplus m$   
return  $(r, x)$



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

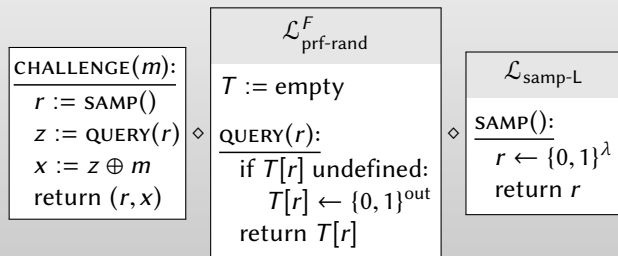
Must use fact that  $r$  is unlikely to repeat (when chosen this way)

# Security proof



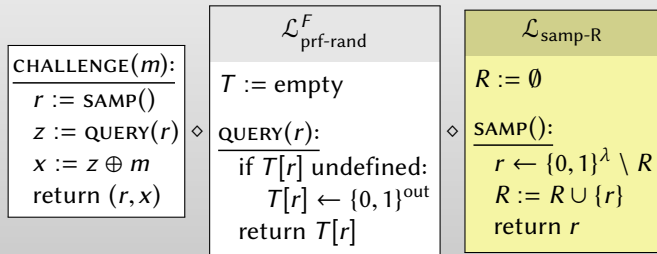
Isolate sampling of  $r$ .

# Security proof



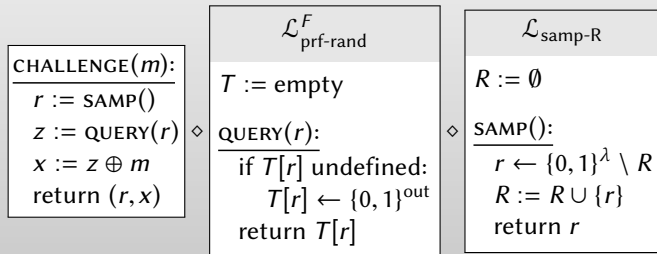
Isolate sampling of  $r$ .

# Security proof



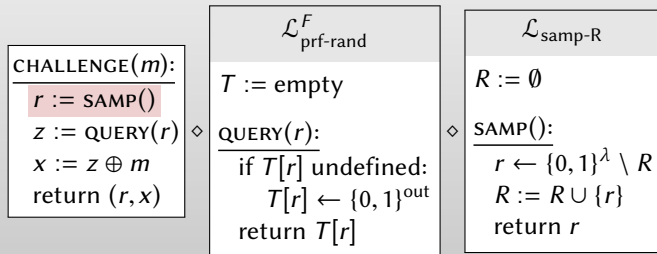
Sample  $r$  without replacement (change  $\mathcal{L}_{\text{samp-L}}$  to  $\mathcal{L}_{\text{samp-R}}$ ).

# Security proof



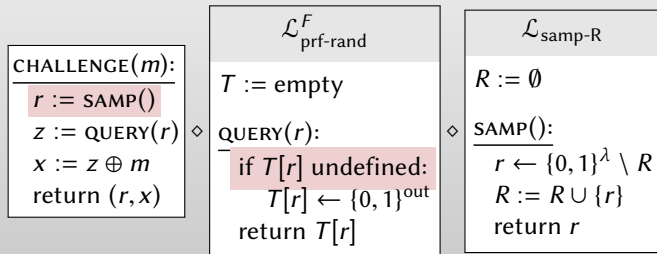
Sample  $r$  without replacement (change  $\mathcal{L}_{\text{samp-L}}$  to  $\mathcal{L}_{\text{samp-R}}$ ).

# Security proof



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

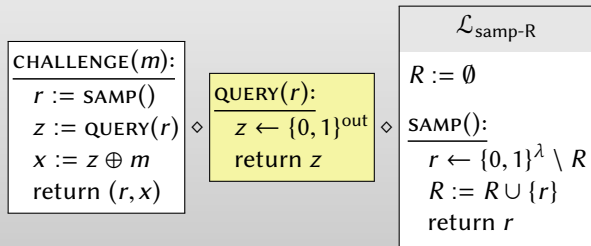
# Security proof



If-statement is always taken.

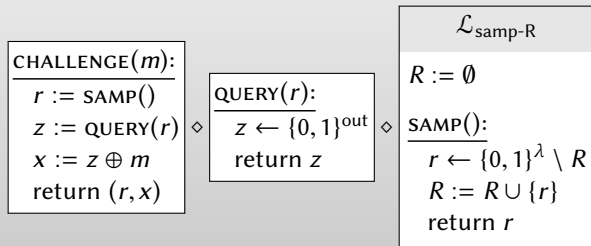


# Security proof



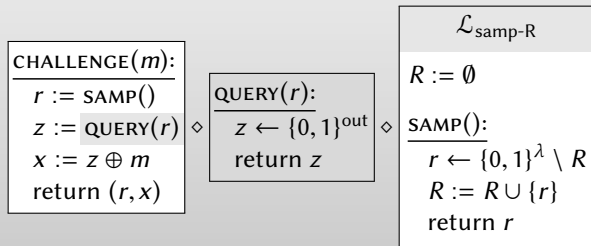
Middle library can therefore be simplified.

# Security proof



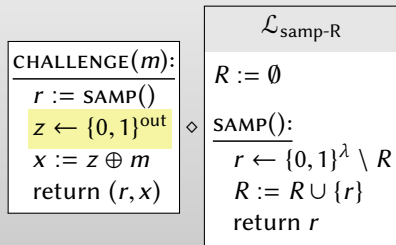
Middle library can therefore be simplified.

# Security proof



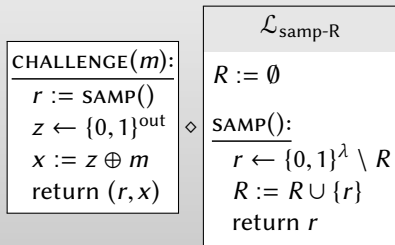
Inline call to QUERY.

# Security proof



Inline call to QUERY.

# Security proof



Inline call to QUERY.

# Security proof



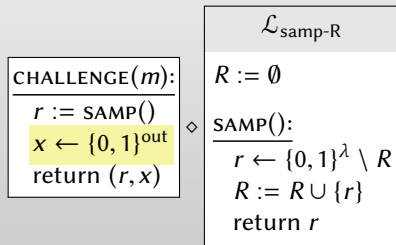
<b>CHALLENGE(<math>m</math>):</b>
$r := \text{SAMP}()$
$z \leftarrow \{0, 1\}^{\text{out}}$
$x := z \oplus m$
return $(r, x)$



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

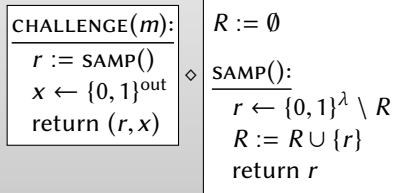
Can apply the “one-time pad rule” (since mask  $z$  is uniform each time)

# Security proof



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

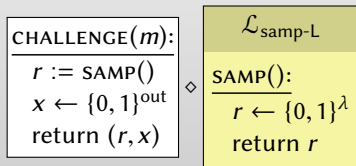
# Security proof



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

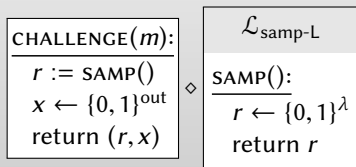


# Security proof



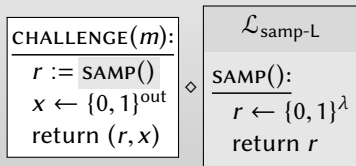
Replace  $\mathcal{L}_{\text{samp-L}}$  with  $\mathcal{L}_{\text{samp-R}}$ .

# Security proof



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

# Security proof



Inline call to SAMP.

# Security proof



CHALLENGE( $m$ ):

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

$x \leftarrow \{0, 1\}^{\text{out}}$

return  $(r, x)$

Inline call to SAMP.

# Security proof



**CHALLENGE( $m$ ):**

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

$x \leftarrow \{0, 1\}^{\text{out}}$

return  $(r, x)$

Inline call to **SAMP**.

# Security proof


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

**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  $\mathcal{L}_{\text{cpa}\text{-rand}}$ .

# Summary

We showed:

$\mathcal{L}_{\text{cpa\$-real}}^\Sigma$		$\mathcal{L}_{\text{cpa\$-rand}}^\Sigma$
$k \leftarrow \{0, 1\}^\lambda$		
<u>CHALLENGE(<math>m</math>):</u>	$\approx$	<u>CHALLENGE(<math>m</math>):</u>
$r \leftarrow \{0, 1\}^\lambda$		$c \leftarrow \{0, 1\}^{\lambda+\text{out}}$
$x := F(k, r) \oplus m$		return $c$
return $(r, x)$		

So our scheme is a CPA\$-secure encryption scheme when  $F$  is a secure PRF.