Cryptography

Lecture 7

Arkady Yerukhimovich

September 18, 2024

Outline

Lecture 6 Review

- Reductions, reductions, reductions
- Security of PRG+OTP
- CPA-Secure Encryption

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Definition: An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ is CPA-secure if for all PPT $\mathcal{A}$ it holds that

$$\Pr[\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + \mathsf{negl}(n)$$

How to Construct CPA-Secure Encryption

- Recall that PRG+OTP encryption allowed us to encrypt long messages.
- But, it still revealed if same message was encrypted many times.

How to Construct CPA-Secure Encryption

- Recall that PRG+OTP encryption allowed us to encrypt long messages.
- But, it still revealed if same message was encrypted many times.

Key Idea

What if encryption (and decryption) could generate a different OTP for each ciphertext?

How to Construct CPA-Secure Encryption

- Recall that PRG+OTP encryption allowed us to encrypt long messages.
- But, it still revealed if same message was encrypted many times.

Key Idea

What if encryption (and decryption) could generate a different OTP for each ciphertext?

Note: We need to produce enough OTP's for as many encryptions as $\mathcal{A}$ wants. So, can't just pre-generate them all.

Outline

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	
0001	
$\vdots$	
1111	

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	
0001	
$\vdots$	
1111	

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	
$\vdots$	
1111	

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	0001
$\vdots$	
1111	

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	0001
$\vdots$	$\vdots$
1111	1101

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	0001
$\vdots$	$\vdots$
1111	1101

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$
- This is the same as choosing a uniformly random function from the set of all n-bit to n-bit functions

Why Random Functions Are Useful for Crypto

- Key feature

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells
    - Thus, there are $|\mathcal{F}_n| = (2^n)^{(2^n)} = 2^{n2^n}$ possible functions

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells
    - Thus, there are $|\mathcal{F}_n| = (2^n)^{(2^n)} = 2^{n2^n}$ possible functions
    - Writing down a random f from $\mathcal{F}_n$ requires $\log |\mathcal{F}_n| = n2^n$ bits

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells
    - Thus, there are $|\mathcal{F}_n| = (2^n)^{(2^n)} = 2^{n2^n}$ possible functions
    - Writing down a random f from $\mathcal{F}_n$ requires $\log|\mathcal{F}_n| = n2^n$ bits

Question:

How can we get the benefits of a random function without paying the overhead?

PRF Goals

Construct an efficient, keyed function
$F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ such that:

PRF Goals

Construct an efficient, keyed function
$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that:

- $F_k(\cdot)$ is efficiently computable

PRF Goals

Construct an efficient, keyed function
$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that:

- $F_k(\cdot)$ is efficiently computable
- For a random key $k \leftarrow \{0,1\}^n$, $F_k(\cdot)$ looks like a *random function* from n bits to n bits (to someone who doesn't know k).

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

Definition: F is a secure PRF if for all PPT distinguishers $\mathcal{D}$, it holds that

$$\Pr[PRF_{\mathcal{D},F}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$$

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

Definition: F is a secure PRF if for all PPT distinguishers $\mathcal{D}$, it holds that

$$\Pr[PRF_{\mathcal{D},F}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$$

$\mathcal{D}$ cannot distinguish between oracle access to a random function and oracle access to a PRF (for a key k he doesn't know).

Observations

Observations:

- $\mathcal{D}$ can make polynomially many queries to $\mathcal{O}$

Observations

Observations:

- $\mathcal{D}$ can make polynomially many queries to $\mathcal{O}$
- $\mathcal{D}$ can choose its queries adaptively based on results of earlier queries

Observations

Observations:

- $\mathcal{D}$ can make polynomially many queries to $\mathcal{O}$
- $\mathcal{D}$ can choose its queries adaptively based on results of earlier queries
- The set of polynomially many evaluations of $F_k(\cdot)$ must look random

Observations

Observations:

- $\mathcal{D}$ can make polynomially many queries to $\mathcal{O}$
- $\mathcal{D}$ can choose its queries adaptively based on results of earlier queries
- The set of polynomially many evaluations of $F_k(\cdot)$ must look random
- Clearly, this is not possible if $\mathcal{D}$ knows k

1) Compute $F_k(0^n) = y_1$

2) Query $\mathcal{O}(0^n) = y_2$

3) if $y_1 = y_2$ output PRF

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.
  - If $\mathcal{O} = F_k$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability 1

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.
  - If $\mathcal{O} = F_k$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability 1
  - If $\mathcal{O} = f$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability $1/2^n$

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.
  - If $\mathcal{O} = F_k$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability 1
  - If $\mathcal{O} = f$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability $1/2^n$
- So, $\mathcal{D}$ always outputs 1 when $\mathcal{O} = F_k$ and outputs 1 with probability $1/2^n$ when $\mathcal{O} = f$.

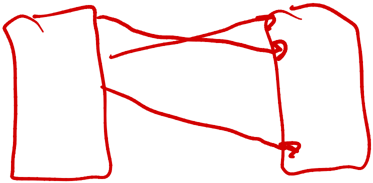
$$\Pr[\mathcal{D}\text{ WINS}] = \Pr[b = 1] \cdot 1 + \Pr[b = 0] \cdot (1 - 1/2^n) > 1/2$$

Variants of PRFs

- Pseudorandom permutation (PRP)

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)



Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP
  - Note that a permutation is always *invertible*. For every permutation f, there is a permutation f^{-1}.

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP
  - Note that a permutation is always *invertible*. For every permutation f, there is a permutation f^{-1}.
  - In a *strong PRP*, we give $\mathcal{D}$ access to oracles for both f and f^{-1}. $\mathcal{D}$ still should not be able to distinguish from a PRP from a random permutation even using both oracles.

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP
  - Note that a permutation is always *invertible*. For every permutation f, there is a permutation f^{-1}.
  - In a *strong PRP*, we give $\mathcal{D}$ access to oracles for both f and f^{-1}. $\mathcal{D}$ still should not be able to distinguish from a PRP from a random permutation even using both oracles.
  - In applied crypto, this is often called a *blockcipher*.

Relationship Between PRG and PRF

Goals:

- Clearly, PRG and PRF have similar goals
- Both construct random-looking objects
- Both use this to “create randomness”

Relationship Between PRG and PRF

Goals:

- Clearly, PRG and PRF have similar goals
- Both construct random-looking objects
- Both use this to “create randomness”

Relationships:

- Not hard to show that a PRF can be used to build a PRG
- In fact, PRG can also be used to build a PRF
- But, important to remember the differences in functionalities and security definitions

Outline

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k,c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k, c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

Intuition:

- $F_k(r)$ serves as a per-ciphertext OTP

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k,c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

Intuition:

- $F_k(r)$ serves as a per-ciphertext OTP
- Need to include r in c to enable decryption, but this is ok since r doesn't reveal anything about m

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k,c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

Intuition:

- $F_k(r)$ serves as a per-ciphertext OTP
- Need to include r in c to enable decryption, but this is ok since r doesn't reveal anything about m
- Only get computational security due to the use of a PRF

CPA-Secure Encryption from a PRF

PRF+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k, c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

Intuition:

- $F_k(r)$ serves as a per-ciphertext OTP
- Need to include r in c to enable decryption, but this is ok since r doesn't reveal anything about m
- Only get computational security due to the use of a PRF

Why Is This Secure?

Consider what happens if we use a random function instead of F_k

Outline

CPA-Secure Encryption from a PRF

PRF+OTP Encryption (Π)

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, F_k(r) \oplus m)$
- $\mathsf{Dec}(k,c)$: Parse c as (r, c'), compute $m = F_k(r) \oplus c'$

Theorem

If F is a secure PRF, then PRF+OTP is CPA-secure

Proof Technique

To prove security from a PRF, we often do the following:

Proof Technique

To prove security from a PRF, we often do the following:

1. Consider the scheme where F_k is replaced by a random function f

Proof Technique

To prove security from a PRF, we often do the following:

1. Consider the scheme where F_k is replaced by a random function f
    - Show by reduction to security of PRF, that $\mathcal{A}$ can't tell we made this change.

Proof Technique

To prove security from a PRF, we often do the following:

1. Consider the scheme where F_k is replaced by a random function f
   - Show by reduction to security of PRF, that $\mathcal{A}$ can't tell we made this change.
   - So, $\mathcal{A}$'s success probability must be (essentially) the same in this and original variant.

Proof Technique

To prove security from a PRF, we often do the following:

1. Consider the scheme where F_k is replaced by a random function f
   - Show by reduction to security of PRF, that $\mathcal{A}$ can't tell we made this change.
   - So, $\mathcal{A}$'s success probability must be (essentially) the same in this and original variant.
2. Use a probabilistic argument to prove that scheme is unconditionally secure when using a random function f.

Proof Technique

To prove security from a PRF, we often do the following:

1. Consider the scheme where F_k is replaced by a random function f
    - Show by reduction to security of PRF, that $\mathcal{A}$ can't tell we made this change.
    - So, $\mathcal{A}$'s success probability must be (essentially) the same in this and original variant.
2. Use a probabilistic argument to prove that scheme is unconditionally secure when using a random function f.
    - Random function is essentially a collection of 2^n OTPs

Proof Technique

To prove security from a PRF, we often do the following:

1. Consider the scheme where F_k is replaced by a random function f
   - Show by reduction to security of PRF, that $\mathcal{A}$ can't tell we made this change.
   - So, $\mathcal{A}$'s success probability must be (essentially) the same in this and original variant.
2. Use a probabilistic argument to prove that scheme is unconditionally secure when using a random function f.
   - Random function is essentially a collection of 2^n OTPs
   - Proof is similar to proof of OTP, but need to account for probability of collision in r

Security of PRF+OTP: Step 1

Define the following encryption scheme $\tilde{\Pi}$:

$\tilde{\Pi}$ Encryption Scheme

- $\widetilde{\mathsf{Gen}}(1^n)$: $f \leftarrow \mathcal{F}_n$ (the set of functions $\{0,1\}^n \to \{0,1\}^n$)
- $\widetilde{\mathsf{Enc}}(k, m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, f(r) \oplus m)$
- $\widetilde{\mathsf{Dec}}(k, c)$: Parse c as (r, c'), compute $m = f(r) \oplus c'$

Security of PRF+OTP: Step 1

Define the following encryption scheme $\tilde{\Pi}$:

$\tilde{\Pi}$ Encryption Scheme

- $\widetilde{\mathsf{Gen}}(1^n)$: $f \leftarrow \mathcal{F}_n$ (the set of functions $\{0,1\}^n \to \{0,1\}^n$)
- $\widetilde{\mathsf{Enc}}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, f(r) \oplus m)$
- $\widetilde{\mathsf{Dec}}(k,c)$: Parse c as (r, c'), compute $m = f(r) \oplus c'$

- Observe that this is exactly PRF+OTP with F_k replaced by f
- This encryption is not efficient as we cannot evaluate a random function
- But, it is useful as a "thought experiment" in the proof as it gives us a target for security

Security of PRF+OTP: Step 1

$\tilde{\Pi}$ Encryption Scheme

- $\widetilde{\mathsf{Gen}}(1^n)$: $f \leftarrow \mathcal{F}_n$ (the set of functions $\{0,1\}^n \to \{0,1\}^n$)
- $\widetilde{\mathsf{Enc}}(k,m)$: Choose $r \leftarrow \{0,1\}^n$, output $c = (r, f(r) \oplus m)$
- $\widetilde{\mathsf{Dec}}(k,c)$: Parse c as (r, c'), compute $m = f(r) \oplus c'$

Lemma: For any PPT $\mathcal{A}$ asking at most $q(n)$ encryption queries

$$\left| \Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \right| \leq \mathrm{negl}(n)$$

Security of PRF+OTP: Proof of Lemma

Lemma

For any PPT $\mathcal{A}$ asking at most $q(n)$ encryption queries

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1]\right| \leq \mathsf{negl}(n)$$

Security of PRF+OTP: Proof of Lemma

Lemma

For any PPT $\mathcal{A}$ asking at most $q(n)$ encryption queries

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1]\right| \leq \mathsf{negl}(n)$$

We prove this lemma by reduction:

Security of PRF+OTP: Proof of Lemma

Lemma

For any PPT $\mathcal{A}$ asking at most $q(n)$ encryption queries

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1]\right| \leq \mathsf{negl}(n)$$

We prove this lemma by reduction:

- Assume there is a PPT $\mathcal{A}_c$ making $q(n)$ queries that distinguishes between Π and $\tilde{\Pi}$

Security of PRF+OTP: Proof of Lemma

Lemma

For any PPT $\mathcal{A}$ asking at most $q(n)$ encryption queries

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1]\right| \leq \mathrm{negl}(n)$$

We prove this lemma by reduction:

- Assume there is a PPT $\mathcal{A}_c$ making $q(n)$ queries that distinguishes between Π and $\tilde{\Pi}$
  - $\mathcal{A}_c$ is a CPA-security adversary
  - What we care about is the difference in probability that $\mathcal{A}_c$ wins the CPA-security game when playing with Π vs. $\tilde{\Pi}$.

Security of PRF+OTP: Proof of Lemma

Lemma

For any PPT $\mathcal{A}$ asking at most $q(n)$ encryption queries

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1]\right| \leq \mathrm{negl}(n)$$

We prove this lemma by reduction:

- Assume there is a PPT $\mathcal{A}_c$ making $q(n)$ queries that distinguishes between Π and $\tilde{\Pi}$
  - $\mathcal{A}_c$ is a CPA-security adversary
  - What we care about is the difference in probability that $\mathcal{A}_c$ wins the CPA-security game when playing with Π vs. $\tilde{\Pi}$.
- Use this to construct $\mathcal{A}_r$ that breaks PRF security of F_k

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b=1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:
  - $\mathcal{A}_r$ is playing the PRF security game

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:
  - $\mathcal{A}_r$ is playing the PRF security game
  - $\mathcal{A}_r$ is given oracle $\mathcal{O} : \{0,1\}^n \rightarrow \{0,1\}^n$ where either $\mathcal{O} = F_k(\cdot)$ (a PRF), or $\mathcal{O} = f(\cdot)$ (a random function)

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:
  - $\mathcal{A}_r$ is playing the PRF security game
  - $\mathcal{A}_r$ is given oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$ where either $\mathcal{O} = F_k(\cdot)$ (a PRF), or $\mathcal{O} = f(\cdot)$ (a random function)
- The CPA-security adversary $\mathcal{A}_c$:

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O}=f$.
  if $b=1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O}=F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n)=1$ (i.e., $\mathcal{D}$ wins) if $b'=b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0|=|m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)=1$ (i.e., $\mathcal{A}$ wins) if $b'=b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:
  - $\mathcal{A}_r$ is playing the PRF security game
  - $\mathcal{A}_r$ is given oracle $\mathcal{O}:\{0,1\}^n \rightarrow \{0,1\}^n$ where either $\mathcal{O}=F_k(\cdot)$ (a PRF), or $\mathcal{O}=f(\cdot)$ (a random function)
- The CPA-security adversary $\mathcal{A}_c$:
  - $\mathcal{A}_c$ plays the CPA-security game against either Π or $\tilde{\Pi}$

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:
  - $\mathcal{A}_r$ is playing the PRF security game
  - $\mathcal{A}_r$ is given oracle $\mathcal{O} : \{0,1\}^n \rightarrow \{0,1\}^n$ where either $\mathcal{O} = F_k(\cdot)$ (a PRF), or $\mathcal{O} = f(\cdot)$ (a random function)
- The CPA-security adversary $\mathcal{A}_c$:
  - $\mathcal{A}_c$ plays the CPA-security game against either Π or $\tilde{\Pi}$
    - to answer encryption queries – The $\mathsf{Enc}(\cdot)$ oracle given to $\mathcal{A}_c$ in Π uses F_k and the oracle in $\tilde{\Pi}$ uses f

The Two Adversaries

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

We have to consider two adversaries, $\mathcal{A}_r$ and $\mathcal{A}_c$

- The PRF adversary $\mathcal{A}_r$:
  - $\mathcal{A}_r$ is playing the PRF security game
  - $\mathcal{A}_r$ is given oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$ where either $\mathcal{O} = F_k(\cdot)$ (a PRF), or $\mathcal{O} = f(\cdot)$ (a random function)
- The CPA-security adversary $\mathcal{A}_c$:
  - $\mathcal{A}_c$ plays the CPA-security game against either Π or $\tilde{\Pi}$
    - to answer encryption queries – The $\mathsf{Enc}(\cdot)$ oracle given to $\mathcal{A}_c$ in Π uses F_k and the oracle in $\tilde{\Pi}$ uses f
  - We care about the *difference* in $\mathcal{A}_c$'s WIN probability

Constructing $\mathcal{A}_r^{\mathcal{O}}$: Intuition

- $\mathcal{A}_r$ needs to use $\mathcal{A}_c$ to win PRF game

Constructing $\mathcal{A}_r^{\mathcal{O}}$: Intuition

- $\mathcal{A}_r$ needs to use $\mathcal{A}_c$ to win PRF game
- $\mathcal{A}_r$ acts as the challenger for $\mathcal{A}_c$ in CPA-security game

Constructing $\mathcal{A}_r^{\mathcal{O}}$: Intuition

- $\mathcal{A}_r$ needs to use $\mathcal{A}_c$ to win PRF game
- $\mathcal{A}_r$ acts as the challenger for $\mathcal{A}_c$ in CPA-security game
    - $\mathcal{A}_r$ must answer $\mathcal{A}_c$'s $\mathsf{Enc}(\cdot)$ queries (i.e., simulate the Enc oracle)
    - $\mathcal{A}_r$ must produce the challenge ciphertext c

Constructing $\mathcal{A}_r^{\mathcal{O}}$: Intuition

- $\mathcal{A}_r$ needs to use $\mathcal{A}_c$ to win PRF game
- $\mathcal{A}_r$ acts as the challenger for $\mathcal{A}_c$ in CPA-security game
    - $\mathcal{A}_r$ must answer $\mathcal{A}_c$'s $\mathrm{Enc}(\cdot)$ queries (i.e., simulate the Enc oracle)
    - $\mathcal{A}_r$ must produce the challenge ciphertext c
- If $\mathcal{A}_c$ WINS, $\mathcal{A}_r$ must use that to win the game against his challenger

Constructing $\mathcal{A}_r^{\mathcal{O}}$

- Run $\mathcal{A}_c(1^n)$ and when $\mathcal{A}_c$ asks $\mathsf{Enc}(m)$ query
  - Choose $r \leftarrow \{0,1\}^n$, query $y = \mathcal{O}(r)$, return $c = (r, y \oplus m)$ to $\mathcal{A}_c$

Constructing $\mathcal{A}_r^{\mathcal{O}}$

- Run $\mathcal{A}_c(1^n)$ and when $\mathcal{A}_c$ asks $\mathsf{Enc}(m)$ query
  - Choose $r \leftarrow \{0,1\}^n$, query $y = \mathcal{O}(r)$, return $c = (r, y \oplus m)$ to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs (m_0, m_1)
  - Choose $b \leftarrow \{0,1\}$
  - Choose $r \leftarrow \{0,1\}^n$, query $y = \mathcal{O}(r)$, return $c = (r, y \oplus m_b)$ as the challenge

Constructing $\mathcal{A}_r^{\mathcal{O}}$

- Run $\mathcal{A}_c(1^n)$ and when $\mathcal{A}_c$ asks $\text{Enc}(m)$ query
  - Choose $r \leftarrow \{0,1\}^n$, query $y = \mathcal{O}(r)$, return $c = (r, y \oplus m)$ to $\mathcal{A}_c$
- When $\mathcal{A}_c$ outputs (m_0, m_1)
  - Choose $b \leftarrow \{0,1\}$
  - Choose $r \leftarrow \{0,1\}^n$, query $y = \mathcal{O}(r)$, return $c = (r, y \oplus m_b)$ as the challenge
- Continue answering Enc queries until $\mathcal{A}_c$ outputs guess b'
  - Output 1 ("PRF") if $b = b'$, and 0 otherwise.

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$
  - This is exactly the CPA-security game vs. Π

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$
  - This is exactly the CPA-security game vs. Π
  - Since $\mathcal{A}_r$ output 1 when $\mathcal{A}_c$ WINS, we have

$$\Pr_{k \leftarrow \{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n) = 1] = \Pr[PrivK^{cpa}_{\mathcal{A}_c,\Pi}(n) = 1]$$

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$
  - This is exactly the CPA-security game vs. Π
  - Since $\mathcal{A}_r$ output 1 when $\mathcal{A}_c$ WINS, we have

$$\Pr_{k \leftarrow \{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n) = 1] = \Pr[PrivK^{cpa}_{\mathcal{A}_c,\Pi}(n) = 1]$$

- Case 2: $\mathcal{O} = f$ (i.e., $b = 0$ in PRF game)

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$
  - This is exactly the CPA-security game vs. Π
  - Since $\mathcal{A}_r$ output 1 when $\mathcal{A}_c$ WINS, we have

$$\Pr_{k \leftarrow \{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n) = 1] = \Pr[PrivK_{\mathcal{A}_c,\Pi}^{cpa}(n) = 1]$$

- Case 2: $\mathcal{O} = f$ (i.e., $b = 0$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $f(r) \oplus m$

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$
  - This is exactly the CPA-security game vs. Π
  - Since $\mathcal{A}_r$ output 1 when $\mathcal{A}_c$ WINS, we have

$$\Pr_{k \leftarrow \{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n) = 1] = \Pr[PrivK_{\mathcal{A}_c,\Pi}^{cpa}(n) = 1]$$

- Case 2: $\mathcal{O} = f$ (i.e., $b = 0$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $f(r) \oplus m$
  - This is exactly the CPA-security game vs. $\tilde{\Pi}$

Analysis of $\mathcal{A}_r$'s success

There are two cases to analyze:

- Case 1: $\mathcal{O} = F_k$ (i.e., $b = 1$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $F_k(r) \oplus m$
  - This is exactly the CPA-security game vs. Π
  - Since $\mathcal{A}_r$ output 1 when $\mathcal{A}_c$ WINS, we have

$$\Pr_{k \leftarrow \{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n) = 1] = \Pr[PrivK^{cpa}_{\mathcal{A}_c,\Pi}(n) = 1]$$

- Case 2: $\mathcal{O} = f$ (i.e., $b = 0$ in PRF game)
  - $\mathcal{A}_r$ answers all Enc queries and produces c with $f(r) \oplus m$
  - This is exactly the CPA-security game vs. $\tilde{\Pi}$
  - Since $\mathcal{A}_r$ output 1 when $\mathcal{A}_c$ WINS, we have

$$\Pr_{f \leftarrow \mathcal{F}_n}[\mathcal{A}_r^{f(\cdot)}(1^n) = 1] = \Pr[PrivK^{cpa}_{\mathcal{A}_c,\tilde{\Pi}}(n) = 1]$$

Analysis of $\mathcal{A}_r$'s success

- We assumed that $\mathcal{A}_c$ distinguishes between Π and $\tilde{\Pi}$

$$\left| \Pr[PrivK^{cpa}_{\mathcal{A}_c,\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A}_c,\tilde{\Pi}}(n) = 1] \right| > 1/\text{poly}(n)$$

Analysis of $\mathcal{A}_r$'s success

- We assumed that $\mathcal{A}_c$ distinguishes between Π and $\tilde{\Pi}$

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A}_c,\Pi}(n)=1]-\Pr[PrivK^{cpa}_{\mathcal{A}_c,\tilde{\Pi}}(n)=1]\right|>1/\text{poly}(n)$$

- By the last slide, this implies that

$$\left|\Pr_{k\leftarrow\{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n)=1]-\Pr_{f\leftarrow\mathcal{F}_n}[\mathcal{A}_r^{f(\cdot)}(1^n)=1]\right|>1/\text{poly}(n)$$

Analysis of $\mathcal{A}_r$'s success

- We assumed that $\mathcal{A}_c$ distinguishes between Π and $\tilde{\Pi}$

$$\left|\Pr[PrivK^{cpa}_{\mathcal{A}_c,\Pi}(n) = 1] - \Pr[PrivK^{cpa}_{\mathcal{A}_c,\tilde{\Pi}}(n) = 1]\right| > 1/\text{poly}(n)$$

- By the last slide, this implies that

$$\left|\Pr_{k \leftarrow \{0,1\}^n}[\mathcal{A}_r^{F_k(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow \mathcal{F}_n}[\mathcal{A}_r^{f(\cdot)}(1^n) = 1]\right| > 1/\text{poly}(n)$$

- That is, $\mathcal{A}_r$ is able to distinguish between $F_k(\cdot)$ and $f(\cdot)$. But, we know that F_k is a PRF.

Contradiction!

Proof Technique

To prove security from a PRF, we often do the following:

✓ Consider the scheme where F_k is replaced by a random function f
  - Show by reduction to security of PRF, that $\mathcal{A}$ can't tell we made this change.
  - So, $\mathcal{A}$'s success probability must be (essentially) the same in this and original variant.

2. Use a probabilistic argument to prove that scheme is unconditionally secure when using a random function f.
   - Random function is essentially a collection of 2^n OTPs
   - Proof is similar to proof of OTP, but need to account for probability of collision in r

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathsf{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n}$$

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathrm{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \le 1/2 + \frac{q(n)}{2^n}$$

- Recall that $\tilde{\Pi}$ encrypts as $c = (r, f(r) \oplus m)$

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathsf{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n}$$

- Recall that $\tilde{\Pi}$ encrypts as $c = (r, f(r) \oplus m)$
- Let r^* be the randomness used to encrypt the challenge

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathrm{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n}$$

- Recall that $\tilde{\Pi}$ encrypts as $c = (r, f(r) \oplus m)$
- Let r^* be the randomness used to encrypt the challenge
- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathrm{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n}$$

- Recall that $\tilde{\Pi}$ encrypts as $c = (r, f(r) \oplus m)$
- Let r^* be the randomness used to encrypt the challenge
- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\mathcal{A}$ knows nothing about $f(r^*)$, so $f(r^*)$ is random (good OTP)
  - $\Pr[\mathcal{A}$ outputs $b = b'] = 1/2$

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathrm{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n}$$

- Recall that $\tilde{\Pi}$ encrypts as $c = (r, f(r) \oplus m)$
- Let r^* be the randomness used to encrypt the challenge
- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\mathcal{A}$ knows nothing about $f(r^*)$, so $f(r^*)$ is random (good OTP)
  - $\Pr[\mathcal{A}$ outputs $b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries

Proving CPA-security of $\tilde{\Pi}$

Lemma

For any $\mathcal{A}$ making at most $q(n)$ queries to $\mathrm{Enc}(\cdot)$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n}$$

- Recall that $\tilde{\Pi}$ encrypts as $c = (r, f(r) \oplus m)$
- Let r^* be the randomness used to encrypt the challenge
- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\mathcal{A}$ knows nothing about $f(r^*)$, so $f(r^*)$ is random (good OTP)
  - $\Pr[\mathcal{A}$ outputs $b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\mathcal{A}$ learns value of $f(r^*)$ (he sees $c = (r^*, c')$, computes $f(r^*) = c' \oplus m$)
  - $\Pr[\mathcal{A}$ outputs $b = b'] = 1$

Proving CPA-security of $\tilde{\Pi}$

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$

Proving CPA-security of $\tilde{\Pi}$

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$

Claim: $\Pr[\text{Case 2}] \leq \text{negl}(n)$

Proving CPA-security of $\tilde{\Pi}$

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$

Claim: $\Pr[\text{Case 2}] \leq \mathsf{negl}(n)$

- We said that $\mathcal{A}$ makes at most $q(n) = \mathsf{poly}(n)$ Enc queries

Proving CPA-security of $\tilde{\Pi}$

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$

Claim: $\Pr[\text{Case 2}] \leq \text{negl}(n)$

- We said that $\mathcal{A}$ makes at most $q(n) = \text{poly}(n)$ Enc queries
- On each Enc query, randomness $r_i \leftarrow \{0, 1\}^n$

Proving CPA-security of $\tilde{\Pi}$

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$

Claim: $\Pr[\text{Case 2}] \leq \mathsf{negl}(n)$

- We said that $\mathcal{A}$ makes at most $q(n) = \mathsf{poly}(n)$ Enc queries
- On each Enc query, randomness $r_i \leftarrow \{0,1\}^n$
- In encrypting challenge, $r^* \leftarrow \{0,1\}^n$

Proving CPA-security of $\tilde{\Pi}$

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$

Claim: $\Pr[\text{Case } 2] \leq \mathsf{negl}(n)$

- We said that $\mathcal{A}$ makes at most $q(n) = \mathsf{poly}(n)$ Enc queries
- On each Enc query, randomness $r_i \leftarrow \{0,1\}^n$
- In encrypting challenge, $r^* \leftarrow \{0,1\}^n$
- So,

$$\Pr[r^* \in \{r_1, \ldots, r_{q(n)}\}] \leq \sum_{i=1}^{q(n)} \Pr[r^* = r_i] = \frac{q(n)}{2^n} \leq \mathsf{negl}(n)$$

Proving CPA-security of $\tilde{\Pi}$: Putting It Together

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$
  - Occurs with probability at most $q(n)/2^n$

Proving CPA-security of $\tilde{\Pi}$: Putting It Together

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$
  - Occurs with probability at most $q(n)/2^n$

$$\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] \;\; = \;\; \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 1}] + \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 2}]$$

Proving CPA-security of $\tilde{\Pi}$: Putting It Together

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$
  - Occurs with probability at most $q(n)/2^n$

$$
\begin{aligned}
\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] &= \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 1}] + \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 2}] \\
&\leq \Pr[\mathcal{A} \text{ WINS} \mid \text{Case 1}] \cdot \Pr[\text{Case 1}] + \Pr[\text{Case 2}]
\end{aligned}
$$

Proving CPA-security of $\tilde{\Pi}$: Putting It Together

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$
  - Occurs with probability at most $q(n)/2^n$

$$\begin{aligned}
\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] &= \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 1}] + \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 2}] \\
&\leq \Pr[\mathcal{A} \text{ WINS} \mid \text{Case 1}] \cdot \Pr[\text{Case 1}] + \Pr[\text{Case 2}] \\
&\leq \Pr[\mathcal{A} \text{ WINS} \mid \text{Case 1}] + \Pr[\text{Case 2}]
\end{aligned}$$

Proving CPA-security of $\tilde{\Pi}$: Putting It Together

- Case 1: r^* is never used when answering $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1/2$
- Case 2: r^* is used when answering one of $\mathcal{A}$'s Enc queries
  - $\Pr[\mathcal{A} \text{ outputs } b = b'] = 1$
  - Occurs with probability at most $q(n)/2^n$

$$
\begin{aligned}
\Pr[PrivK^{cpa}_{\mathcal{A},\tilde{\Pi}}(n) = 1] &= \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 1}] + \Pr[\mathcal{A} \text{ WINS } \wedge \text{ Case 2}] \\
&\leq \Pr[\mathcal{A} \text{ WINS} \mid \text{Case 1}] \cdot \Pr[\text{Case 1}] + \Pr[\text{Case 2}] \\
&\leq \Pr[\mathcal{A} \text{ WINS} \mid \text{Case 1}] + \Pr[\text{Case 2}] \\
&\leq 1/2 + \frac{q(n)}{2^n}
\end{aligned}
$$

Finishing Proof of CPA-security of PRF+OTP

- ✓ Consider the scheme where F_k is replaced by a random function f
  - We showed that any PPT $\mathcal{A}$ has only a $\mathsf{negl}(n)$ advantage in distinguishing the two games
- ✓ Use a probabilistic argument to prove that scheme is unconditionally secure when using a random function f.
  - We showed that PPT $\mathcal{A}$ WINS with probability $\leq 1/2 + q(n)/2^n$

Finishing Proof of CPA-security of PRF+OTP

- ✓ Consider the scheme where F_k is replaced by a random function f
  - We showed that any PPT $\mathcal{A}$ has only a negl(n) advantage in distinguishing the two games
- ✓ Use a probabilistic argument to prove that scheme is unconditionally secure when using a random function f.
  - We showed that PPT $\mathcal{A}$ WINS with probability $\leq 1/2 + q(n)/2^n$

Combining these two statements, we get that for any PPT $\mathcal{A}$,

$$\Pr[PrivK^{cpa}_{\mathcal{A},\text{PRF+OTP}}(n) = 1] \leq 1/2 + \frac{q(n)}{2^n} + \text{negl}(n)$$