

# THE FUNDAMENTAL LEMMA OF GAME PLAYING

ADVANCED TOPICS IN ~~CYBERSECURITY~~ CRYPTOGRAPHY (7CCSMATC)

---

Martin R. Albrecht

# OUTLINE

Introduction

CTR Mode

Fundamental Lemma of Game Playing

# INTRODUCTION

---

## RECAP

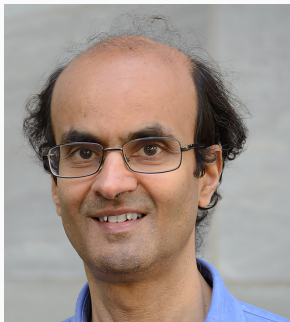
- We have defined what it means for an encryption scheme to be secure (IND-CPA + INT-CTXT = IND-CCA).
- We have shown that the OTP achieves IND-CPA security, even **unconditionally**.

The One-Time Pad is impractical, we want something more manageable  $\Rightarrow$   
Pseudorandomness!

## MAIN REFERENCE

Mihir Bellare and Phillip Rogaway. **Code-Based Game-Playing Proofs and the Security of Triple Encryption**. Cryptology ePrint Archive, Report 2004/331. 2004. URL: <https://eprint.iacr.org/2004/331>

---



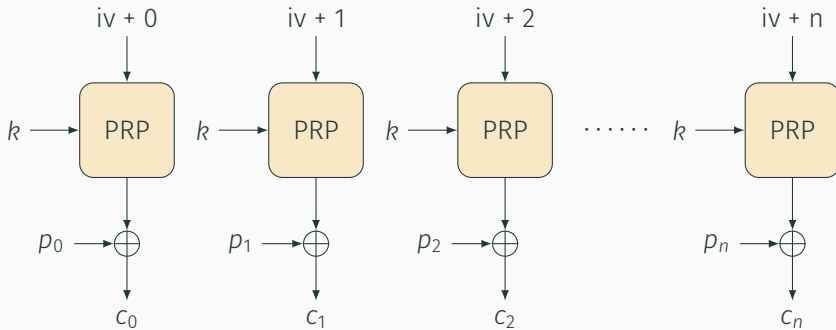
Mihir Bellare is a professor at UCSD

- 2003** RSA Conference's Sixth Annual Award
- 2013** Fellow of the Association for Computing Machinery.
- 2019** Levchin Prize for Real-World Cryptography

CTR MODE

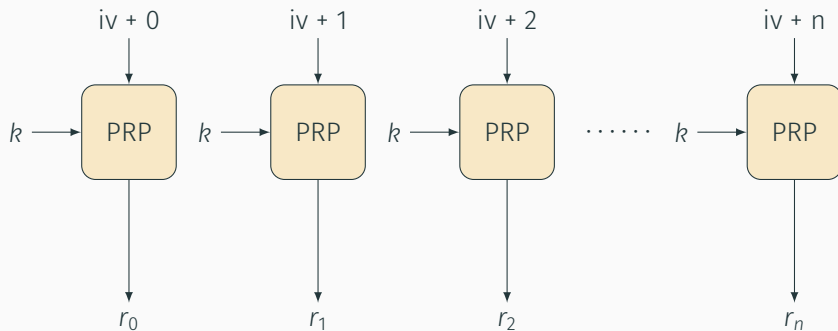
---

# CTR MODE



Picture credit: <https://www.iacr.org/authors/tikz/>

## CTR MODE STREAM



$$r_i \in \{0, 1\}^\lambda$$



## WANT: $n + 1$ PSEUDORANDOM STRINGS OF LENGTH $\lambda$

### Definition (PRF)

A PRF is a keyed function  $F_k : \{0, 1\}^\lambda \rightarrow \{0, 1\}^N$  where  $N$  depends on  $\lambda$  and for  $k \leftarrow \$ \mathcal{K}$ . We say  $F_k$  is  $(t, \varepsilon)$ -secure **PRF** if for  $\text{Game}_0$  and  $\text{Game}_1$  defined below we have:

$$\forall \mathcal{D} \in t \text{ steps: } \text{Adv}_F^{\text{prf}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]| < \varepsilon$$

Game <sub>0</sub>	F(x)
1: $f \leftarrow \emptyset$	1: <b>if</b> $x \notin f.\text{keys}$ <b>then</b> $f[x] \leftarrow \$ \{0, 1\}^N$
2: <b>return</b> $\mathcal{D}^f$	2: $y \leftarrow f[x]$
Game <sub>1</sub>	3: $y \leftarrow F_k(x)$ // Game <sub>1</sub>
1: $f \leftarrow \emptyset; k \leftarrow \$ \mathcal{K}$	4: <b>return</b> $y$
2: <b>return</b> $\mathcal{D}^f$	

## HAVE: $n + 1$ CALLS TO PSEUDORANDOM PERMUTATION OF LENGTH $\lambda$

### Definition (PRP)

A PRP is a keyed permutation  $E_k : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$  for  $k \leftarrow \$ \mathcal{K}$ . We say  $E$  is  $(t, \varepsilon)$ -secure PRP if for  $\text{Game}_0$  and  $\text{Game}_1$  defined below we have:

$$\forall \mathcal{D} \in t \text{ steps: } \text{Adv}_E^{\text{prp}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]| < \varepsilon$$

Game <sub>0</sub>	P(x)
1: $f \leftarrow \emptyset$	1: <b>if</b> $x \notin f.\text{keys}$ <b>then</b> $f[x] \leftarrow \$ \{0, 1\}^\lambda \setminus f.\text{values}$
2: <b>return</b> $\mathcal{D}^P$	2: $y \leftarrow f[x]$
Game <sub>1</sub>	3: $y \leftarrow E_k(x)$ //Game <sub>1</sub>
1: $f \leftarrow \emptyset; k \leftarrow \$ \mathcal{K}$	4: <b>return</b> $y$
2: <b>return</b> $\mathcal{D}^P$	

Game <sub>0</sub>	F(x)
1: $f \leftarrow \emptyset$	1: <b>if</b> $x \in f.\text{keys}$ <b>then</b>
2: <b>return</b> $\mathcal{D}^F$	2: $y \leftarrow f[x]$
Game <sub>1</sub>	3: <b>else</b>
1: $f \leftarrow \emptyset;$	4: $y \leftarrow \$ \{0, 1\}^\lambda \setminus f.\text{values}$
2: <b>return</b> $\mathcal{D}^F$	5: $y \leftarrow \$ \{0, 1\}^\lambda$ // Game <sub>1</sub>
	6: $f[x] \leftarrow y$
	7: <b>return</b> $y$

## PRP-PRF SWITCHING LEMMA

### Lemma

Let  $\pi$  be a random **permutation** from  $\{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ ; let  $\rho$  be a random **function** from  $\{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ . Let  $\mathcal{A}$  be an adversary making at most  $q$  queries to its oracle, then:

$$|\Pr[\mathcal{A}^\pi] - \Pr[\mathcal{A}^\rho]| \leq \frac{q \cdot (q-1)}{2^{\lambda+1}}.$$

# PRP-PRF SWITCHING LEMMA I

Consider the following games:

Game <sub>0</sub>	P(x)
1: $\pi \leftarrow \emptyset$	1: <b>if</b> $x \in \pi.\text{keys}$ <b>then return</b> $\pi[x]$
2: <b>return</b> $\mathcal{A}^P$	2: $y \leftarrow_{\$} \{0, 1\}^\lambda$
Game <sub>1</sub>	3: <b>if</b> $y \in \pi.\text{values}$ <b>then</b>
1: $\pi \leftarrow \emptyset$	4: $\text{bad} \leftarrow \text{true}$
2: <b>return</b> $\mathcal{A}^P$	5: $y \leftarrow_{\$} \{0, 1\}^\lambda \setminus \pi.\text{values}$ // Game <sub>1</sub>
	6: $\pi[x] \leftarrow y$
	7: <b>return</b> $y$

## PRP-PRF SWITCHING LEMMA II

$$|\Pr[\mathcal{A}^\pi] - \Pr[\mathcal{A}^\rho]| = |\Pr[\mathcal{A}^{\text{Game}_0}] - \Pr[\mathcal{A}^{\text{Game}_1}]| \quad (1)$$

$$\leq \Pr[\mathcal{A}^{\text{Game}_0}] \text{ sets bad} \quad (2)$$

$$\leq q \cdot (q + 1)/2^{\lambda+1} \quad (3)$$

**On Eq. (1):**  $\text{Game}_0$  perfectly simulates a random function  $\rho$  and  $\text{Game}_1$  perfectly simulates a random permutation  $\pi$ , by the **principle of lazy sampling**.

Thus, we have

$$\Pr[\mathcal{A}^\rho] = \Pr[\mathcal{A}^{\text{Game}_0}] \text{ and } \Pr[\mathcal{A}^{\text{Game}_1}] = \Pr[\mathcal{A}^\pi].$$

**On Eq. (2):** we will appeal to the **fundamental lemma of game playing**.

**On Eq. (3):** by the union bound the probability that  $y \in \pi.\text{values}$ , is at most

$$\frac{(1 + 2 + \cdots + (q - 1))}{2^\lambda} = \frac{q \cdot (q - 1)}{2^{\lambda+1}}.$$

# FUNDAMENTAL LEMMA OF GAME PLAYING

---

We say  $\text{Game}_0$  and  $\text{Game}_1$  are “identical-until-bad” if they are ... identical until some flag bad is set.



# FUNDAMENTAL LEMMA OF GAME PLAYING

## Lemma (Fundamental Lemma of Game Playing)

*Let  $\text{Game}_0$ ,  $\text{Game}_1$ ,  $\text{Game}_2$  be identical-until-bad games and  $\mathcal{A}$  be an adversary. Then*

$$\begin{aligned} |\Pr[\mathcal{A}^{\text{Game}_0}] - \Pr[\mathcal{A}^{\text{Game}_1}]| &\leq \Pr[\mathcal{A}^{\text{Game}_2} \text{ sets bad}] \text{ and} \\ |\Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_1^{\mathcal{A}}]| &\leq \Pr[\text{Game}_2^{\mathcal{A}} \text{ sets bad}]. \end{aligned}$$

- The first statement follows immediately from the second.
- For the second statement we first prove it with  $\text{Game}_2 = \text{Game}_0$  and then generalise.

## (OPTIONAL) PROOF I

We require that both the adversary and the game always terminate in finite time.

- For any adversary  $\mathcal{A}$  there must exist an integer  $T$  such that  $\mathcal{A}$  always halts within  $T$  steps (regardless of the random choices  $\mathcal{A}$  makes and the answers it receives to its oracle queries).
- For any game  $\text{Game}$  there must exist an integer  $T$  such that  $\text{Game}$  always halts within  $T$  steps (regardless of the random choices made).

## (OPTIONAL) PROOF II

Since  $\mathcal{A}$  and Game terminate in finite time,

- there must be an integer  $T$  such that they each execute at most  $T$  random-assignment statements, and
- there must be an integer  $B$  such that the size of the set  $\mathcal{S}$  in any random-assignment statement  $s \leftarrow \$ \mathcal{S}$  executed by the adversary or the game is at most  $B$ .

$\Rightarrow$  The execution of Game with  $\mathcal{A}$  uses finite randomness, meaning Game and  $\mathcal{A}$  are underlain by a finite sample space  $\Omega$ .

### Punchline

Probabilities are well-defined and we can talk about the probabilities of various events in the execution.

## (OPTIONAL) PROOF III

- This means that there exists an integer  $z$  such that the execution of  $\text{Game}_0$  with  $\mathcal{A}$  and the execution of  $\text{Game}_1$  with  $\mathcal{A}$  perform no more than  $z$  random-assignment statements, each of these sampling from a set of size at most  $z$ .

## (OPTIONAL) PROOF IV

- Let  $\mathcal{C} := \text{Coins}(\mathcal{A}, \text{Game}_0, \text{Game}_1) = [1 \dots z!]^z$  be the set of  $z$ -tuples of numbers, each number between 0 and  $z!$ .

```
z = 2
R = IntegerModRing(factorial(z)); offset = vector(R, z, [1]*z).lift()
Coins = [coin.lift() + offset for coin in FreeModule(R, z)]
print(Coins)
```

[(1, 1), (2, 1), (1, 2), (2, 2)]

- For  $\mathbf{c} = (c_0, \dots, c_{z-1}) \in \mathcal{C}$ , the execution of Game with  $\mathcal{A}$  on coins  $\mathbf{c}$  is defined as follows:
  - On the  $i$ -th random-assignment statement, call it  $x \leftarrow \mathcal{U}(\mathcal{S})$ , where  $\mathcal{S} := \{s_i\}_{0 \leq i < m}$ , if  $\mathcal{S} \neq \emptyset$ , return  $s_{c_i \bmod |\mathcal{S}|}$ , otherwise return  $\perp$ .
- This way to perform random-assignment statements is done regardless of whether it is  $\mathcal{A}$  or one of the procedures from Game that is performing the random-assignment statement.

## (OPTIONAL) PROOF V

- Note that  $m = |\mathcal{S}|$  satisfies  $m|z|$  so if  $\mathbf{c}$  is chosen at random from  $\mathcal{C}$  then the mechanism above will return a point  $x$  drawn uniformly from  $\mathcal{S}$ , and also the values for each random-assignment statement are independent.

## (OPTIONAL) PROOF VI

- For  $\mathbf{c} \in \mathcal{C}$  we let  $\text{Game}_0^{\mathcal{A}}(\mathbf{c})$  denote the output of  $\text{Game}_0$  when  $\text{Game}_0$  is executed with  $\mathcal{A}$  on coins  $\mathbf{c}$ . Same for  $\text{Game}_1$ .
- Write  $\mathcal{C}_{i,\text{one}} := \{\mathbf{c} \in \mathcal{C} : \text{Game}_i^{\mathcal{A}}(\mathbf{c}) \Rightarrow 1\}$
- Write  $\mathcal{C}_i^{\text{bad}} \subseteq \mathcal{C}$  for the coins that result in *bad* being set to **true** when running  $\text{Game}_i^{\mathcal{A}}$ .
- Partition  $\mathcal{C}_{i,\text{one}}$  into  $\mathcal{C}_{i,\text{one}}^{\text{bad}}$  and  $\mathcal{C}_{i,\text{one}}^{\text{good}}$  depending on whether *bad* was set or not in game  $\text{Game}_i$ .
- Because games  $\text{Game}_0$  and  $\text{Game}_1$  are identical-until-*bad*, an element  $\mathbf{c} \in \mathcal{C}$  is in  $\mathcal{C}_{0,\text{one}}^{\text{good}}$  if and only if it is in  $\mathcal{C}_{1,\text{one}}^{\text{good}}$ .
  - *Bad* is never set so the sets are same and in particular have the same size.

## (OPTIONAL) PROOF VII

We then get:

$$\begin{aligned}\Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_1^{\mathcal{A}}] &= \frac{c_{0,\text{one}}}{c} - \frac{c_{1,\text{one}}}{c} \\ &= \frac{c_{0,\text{one}}^{\text{good}} + c_{0,\text{one}}^{\text{bad}}}{c} - \frac{c_{1,\text{one}}^{\text{good}} + c_{1,\text{one}}^{\text{bad}}}{c} \\ &= \frac{c_{0,\text{one}}^{\text{bad}}}{c} - \frac{c_{1,\text{one}}^{\text{bad}}}{c} \\ &\leq \frac{c_{0,\text{one}}^{\text{bad}}}{c} \\ &\leq \frac{c_0^{\text{bad}}}{c} \\ &= \Pr[\text{Game}_0^{\mathcal{A}} \text{ sets bad}].\end{aligned}$$



To prove the second statement we rely on the following lemma.

### Lemma

*Let  $\text{Game}_0$  and  $\text{Game}_1$  be identical-until-bad games. Let  $\mathcal{A}$  be an adversary. Then*

$$\Pr[\text{Game}_0^{\mathcal{A}} \text{ sets bad}] = \Pr[\text{Game}_1^{\mathcal{A}} \text{ sets bad}].$$

## (OPTIONAL) PROOF IX

- Since  $\text{Game}_0$  and  $\text{Game}_1$  are identical-until-bad, each  $c \in \mathcal{C}$  causes bad to be set in  $\text{Game}_0^{\mathcal{A}}$  if and only if it is set in  $\text{Game}_1^{\mathcal{A}}$ .
- Thus

$$\mathcal{C}_1^{bad} = \mathcal{C}_2^{bad}$$

$$|\mathcal{C}_1^{bad}| = |\mathcal{C}_2^{bad}|$$

$$|\mathcal{C}_1^{bad}|/|\mathcal{C}| = |\mathcal{C}_2^{bad}|/|\mathcal{C}|$$

$$\Pr[\text{Game}_1^{\mathcal{A}} \text{ sets bad}] = \Pr[\text{Game}_2^{\mathcal{A}} \text{ sets bad}].$$

# MATCHING ATTACK

- Call  $\sqrt{2^\lambda} = 2^{\lambda/2}$  times and check if any answer repeats.
- By the birthday bound this happens with constant probability

## Memory-less Attack

Read about the Pollard-rho attack to learn how to make this attack use  $\text{poly}(\lambda)$  memory instead of  $2^{\lambda/2}$ .

FIN

WE WANT TO APPROXIMATE THE ONE-TIME PAD

IF WE HAVE A PRF, THIS IS STRAIGHT-FORWARD

IF WE “ONLY” HAVE A PRP, **AN IDEAL PRIMITIVE**, THIS BREAKS  
DOWN AFTER  $q = \sqrt{2^\lambda}$  QUERIES, E.G.  $2^{64}$  FOR  $\lambda = 128$  (AES-128).

**NEXT:** HOW DO WE GET A PRP?

- [BR04] Mihir Bellare and Phillip Rogaway. **Code-Based Game-Playing Proofs and the Security of Triple Encryption**. Cryptology ePrint Archive, Report 2004/331. 2004. URL: <https://eprint.iacr.org/2004/331>.