

# Roadmap

- ❑ Getting rid of the first restriction imposed by perfect security
  - ❖ Encrypting long messages using short keys
- ❑ Pseudorandom generators
  - ❖ Various equivalent definitions

# Encrypting Long Messages Using Short Keys : The Basic Idea

$$\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0, 1\}^L \quad G: \{0, 1\}^\ell \rightarrow \{0, 1\}^L, \ell < L$$

$m \in \mathcal{M}$



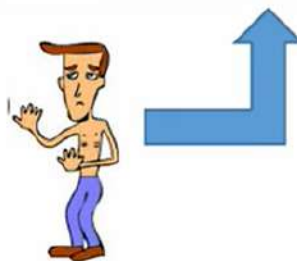
$s \in_r \{0, 1\}^\ell$

$$c := m \oplus G(s)$$



$s \in_r \{0, 1\}^\ell$

Computationally  
bounded



- A **computationally unbounded** adversary cannot distinguish whether  $c$  is an encryption of  $m_0$  or  $m_1$ , since the **pad  $k$**  is a uniformly random  $L$ -bit string

# Encrypting Long Messages Using Short Keys : The Basic Idea

$$\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0, 1\}^L \quad G: \{0, 1\}^\ell \rightarrow \{0, 1\}^L, \ell < L$$

$m \in \mathcal{M}$



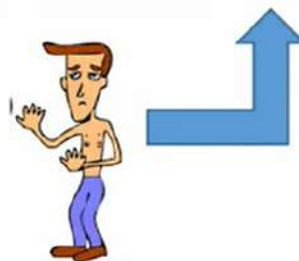
$s \in_r \{0, 1\}^\ell$

$$c := m \oplus G(s)$$



$s \in_r \{0, 1\}^\ell$

Computationally  
bounded



□ A **computationally bounded** adversary cannot distinguish between  $G(s)$  and a uniformly random string from  $\{0, 1\}^L$



A **computationally bounded** adversary cannot distinguish whether  $c$  is an encryption of  $m_0$  or  $m_1$

# Pseudorandom Generator (PRG)

Algorithm  $G$

```

1  Algorithm 7.41: Reed-Lucassen Method for NTRU
2  Input:  $\ell \in \mathbb{N}$ , the bit length of the output
3  Output:  $G(s) \in \{0,1\}^L$ 
4  Let  $s \in_r \{0,1\}^\ell$ 
5  Let  $p = 3$ ,  $q = 2^{\ell+1}$ , and  $N = 2^{\ell+1}$ 
6  Let  $r = 1, \dots, \ell$ 
7  For  $j = 1$  to  $\ell$ 
8  Let  $s_j = s \bmod 2$ 
9  Let  $s = (s - s_j) / 2$ 
10 Let  $s = (s + s_j) / 2$ 
11 Let  $s = (s + s_j) / 2$ 
12 Let  $s = (s + s_j) / 2$ 
13 Let  $s = (s + s_j) / 2$ 
14 Let  $s = (s + s_j) / 2$ 
15 Let  $s = (s + s_j) / 2$ 
16 Let  $s = (s + s_j) / 2$ 
17 Let  $s = (s + s_j) / 2$ 
18 Let  $s = (s + s_j) / 2$ 
19 Let  $s = (s + s_j) / 2$ 
20 Let  $s = (s + s_j) / 2$ 
21 Let  $s = (s + s_j) / 2$ 
22 Let  $s = (s + s_j) / 2$ 
23 Let  $s = (s + s_j) / 2$ 
24 Let  $s = (s + s_j) / 2$ 
25 Let  $s = (s + s_j) / 2$ 
26 Let  $s = (s + s_j) / 2$ 
27 Let  $s = (s + s_j) / 2$ 
28 Let  $s = (s + s_j) / 2$ 
29 Let  $s = (s + s_j) / 2$ 
30 Let  $s = (s + s_j) / 2$ 
31 Let  $s = (s + s_j) / 2$ 
32 Let  $s = (s + s_j) / 2$ 
33 Let  $s = (s + s_j) / 2$ 
34 Let  $s = (s + s_j) / 2$ 
35 Let  $s = (s + s_j) / 2$ 
36 Let  $s = (s + s_j) / 2$ 
37 Let  $s = (s + s_j) / 2$ 
38 Let  $s = (s + s_j) / 2$ 
39 Let  $s = (s + s_j) / 2$ 
40 Let  $s = (s + s_j) / 2$ 
41 Let  $s = (s + s_j) / 2$ 
42 Let  $s = (s + s_j) / 2$ 
43 Let  $s = (s + s_j) / 2$ 
44 Let  $s = (s + s_j) / 2$ 
45 Let  $s = (s + s_j) / 2$ 
46 Let  $s = (s + s_j) / 2$ 
47 Let  $s = (s + s_j) / 2$ 
48 Let  $s = (s + s_j) / 2$ 
49 Let  $s = (s + s_j) / 2$ 
50 Let  $s = (s + s_j) / 2$ 
51 Let  $s = (s + s_j) / 2$ 
52 Let  $s = (s + s_j) / 2$ 
53 Let  $s = (s + s_j) / 2$ 
54 Let  $s = (s + s_j) / 2$ 
55 Let  $s = (s + s_j) / 2$ 
56 Let  $s = (s + s_j) / 2$ 
57 Let  $s = (s + s_j) / 2$ 
58 Let  $s = (s + s_j) / 2$ 
59 Let  $s = (s + s_j) / 2$ 
60 Let  $s = (s + s_j) / 2$ 
61 Let  $s = (s + s_j) / 2$ 
62 Let  $s = (s + s_j) / 2$ 
63 Let  $s = (s + s_j) / 2$ 
64 Let  $s = (s + s_j) / 2$ 
65 Let  $s = (s + s_j) / 2$ 
66 Let  $s = (s + s_j) / 2$ 
67 Let  $s = (s + s_j) / 2$ 
68 Let  $s = (s + s_j) / 2$ 
69 Let  $s = (s + s_j) / 2$ 
70 Let  $s = (s + s_j) / 2$ 
71 Let  $s = (s + s_j) / 2$ 
72 Let  $s = (s + s_j) / 2$ 
73 Let  $s = (s + s_j) / 2$ 
74 Let  $s = (s + s_j) / 2$ 
75 Let  $s = (s + s_j) / 2$ 
76 Let  $s = (s + s_j) / 2$ 
77 Let  $s = (s + s_j) / 2$ 
78 Let  $s = (s + s_j) / 2$ 
79 Let  $s = (s + s_j) / 2$ 
80 Let  $s = (s + s_j) / 2$ 
81 Let  $s = (s + s_j) / 2$ 
82 Let  $s = (s + s_j) / 2$ 
83 Let  $s = (s + s_j) / 2$ 
84 Let  $s = (s + s_j) / 2$ 
85 Let  $s = (s + s_j) / 2$ 
86 Let  $s = (s + s_j) / 2$ 
87 Let  $s = (s + s_j) / 2$ 
88 Let  $s = (s + s_j) / 2$ 
89 Let  $s = (s + s_j) / 2$ 
90 Let  $s = (s + s_j) / 2$ 
91 Let  $s = (s + s_j) / 2$ 
92 Let  $s = (s + s_j) / 2$ 
93 Let  $s = (s + s_j) / 2$ 
94 Let  $s = (s + s_j) / 2$ 
95 Let  $s = (s + s_j) / 2$ 
96 Let  $s = (s + s_j) / 2$ 
97 Let  $s = (s + s_j) / 2$ 
98 Let  $s = (s + s_j) / 2$ 
99 Let  $s = (s + s_j) / 2$ 
100 Let  $s = (s + s_j) / 2$ 

```

Deterministic

$$G(s) \in \{0,1\}^L$$

TRG  $G'$

```

1  Algorithm 7.41: Reed-Lucassen Method for NTRU
2  Input:  $\ell \in \mathbb{N}$ , the bit length of the output
3  Output:  $R \in_r \{0,1\}^L$ 
4  Let  $s \in_r \{0,1\}^\ell$ 
5  Let  $p = 3$ ,  $q = 2^{\ell+1}$ , and  $N = 2^{\ell+1}$ 
6  Let  $r = 1, \dots, \ell$ 
7  For  $j = 1$  to  $\ell$ 
8  Let  $s_j = s \bmod 2$ 
9  Let  $s = (s - s_j) / 2$ 
10 Let  $s = (s + s_j) / 2$ 
11 Let  $s = (s + s_j) / 2$ 
12 Let  $s = (s + s_j) / 2$ 
13 Let  $s = (s + s_j) / 2$ 
14 Let  $s = (s + s_j) / 2$ 
15 Let  $s = (s + s_j) / 2$ 
16 Let  $s = (s + s_j) / 2$ 
17 Let  $s = (s + s_j) / 2$ 
18 Let  $s = (s + s_j) / 2$ 
19 Let  $s = (s + s_j) / 2$ 
20 Let  $s = (s + s_j) / 2$ 
21 Let  $s = (s + s_j) / 2$ 
22 Let  $s = (s + s_j) / 2$ 
23 Let  $s = (s + s_j) / 2$ 
24 Let  $s = (s + s_j) / 2$ 
25 Let  $s = (s + s_j) / 2$ 
26 Let  $s = (s + s_j) / 2$ 
27 Let  $s = (s + s_j) / 2$ 
28 Let  $s = (s + s_j) / 2$ 
29 Let  $s = (s + s_j) / 2$ 
30 Let  $s = (s + s_j) / 2$ 
31 Let  $s = (s + s_j) / 2$ 
32 Let  $s = (s + s_j) / 2$ 
33 Let  $s = (s + s_j) / 2$ 
34 Let  $s = (s + s_j) / 2$ 
35 Let  $s = (s + s_j) / 2$ 
36 Let  $s = (s + s_j) / 2$ 
37 Let  $s = (s + s_j) / 2$ 
38 Let  $s = (s + s_j) / 2$ 
39 Let  $s = (s + s_j) / 2$ 
40 Let  $s = (s + s_j) / 2$ 
41 Let  $s = (s + s_j) / 2$ 
42 Let  $s = (s + s_j) / 2$ 
43 Let  $s = (s + s_j) / 2$ 
44 Let  $s = (s + s_j) / 2$ 
45 Let  $s = (s + s_j) / 2$ 
46 Let  $s = (s + s_j) / 2$ 
47 Let  $s = (s + s_j) / 2$ 
48 Let  $s = (s + s_j) / 2$ 
49 Let  $s = (s + s_j) / 2$ 
50 Let  $s = (s + s_j) / 2$ 
51 Let  $s = (s + s_j) / 2$ 
52 Let  $s = (s + s_j) / 2$ 
53 Let  $s = (s + s_j) / 2$ 
54 Let  $s = (s + s_j) / 2$ 
55 Let  $s = (s + s_j) / 2$ 
56 Let  $s = (s + s_j) / 2$ 
57 Let  $s = (s + s_j) / 2$ 
58 Let  $s = (s + s_j) / 2$ 
59 Let  $s = (s + s_j) / 2$ 
60 Let  $s = (s + s_j) / 2$ 
61 Let  $s = (s + s_j) / 2$ 
62 Let  $s = (s + s_j) / 2$ 
63 Let  $s = (s + s_j) / 2$ 
64 Let  $s = (s + s_j) / 2$ 
65 Let  $s = (s + s_j) / 2$ 
66 Let  $s = (s + s_j) / 2$ 
67 Let  $s = (s + s_j) / 2$ 
68 Let  $s = (s + s_j) / 2$ 
69 Let  $s = (s + s_j) / 2$ 
70 Let  $s = (s + s_j) / 2$ 
71 Let  $s = (s + s_j) / 2$ 
72 Let  $s = (s + s_j) / 2$ 
73 Let  $s = (s + s_j) / 2$ 
74 Let  $s = (s + s_j) / 2$ 
75 Let  $s = (s + s_j) / 2$ 
76 Let  $s = (s + s_j) / 2$ 
77 Let  $s = (s + s_j) / 2$ 
78 Let  $s = (s + s_j) / 2$ 
79 Let  $s = (s + s_j) / 2$ 
80 Let  $s = (s + s_j) / 2$ 
81 Let  $s = (s + s_j) / 2$ 
82 Let  $s = (s + s_j) / 2$ 
83 Let  $s = (s + s_j) / 2$ 
84 Let  $s = (s + s_j) / 2$ 
85 Let  $s = (s + s_j) / 2$ 
86 Let  $s = (s + s_j) / 2$ 
87 Let  $s = (s + s_j) / 2$ 
88 Let  $s = (s + s_j) / 2$ 
89 Let  $s = (s + s_j) / 2$ 
90 Let  $s = (s + s_j) / 2$ 
91 Let  $s = (s + s_j) / 2$ 
92 Let  $s = (s + s_j) / 2$ 
93 Let  $s = (s + s_j) / 2$ 
94 Let  $s = (s + s_j) / 2$ 
95 Let  $s = (s + s_j) / 2$ 
96 Let  $s = (s + s_j) / 2$ 
97 Let  $s = (s + s_j) / 2$ 
98 Let  $s = (s + s_j) / 2$ 
99 Let  $s = (s + s_j) / 2$ 
100 Let  $s = (s + s_j) / 2$ 

```

Randomized

$$R \in_r \{0,1\}^L$$

Requirements from algorithm  $G$ :

- ❖  $G$  should be an **efficient algorithm**
- ❖ **Expansion** :  $L > \ell$
- ❖ **Pseudo randomness (informal)**: no efficient **statistical test** should **significantly separate apart** an output of  $G$  from the output of an  $L$ -bit truly random generator (TRG)



# Indistinguishability Based Definition of PRG

### Algorithm $G$

$$s \in_r \{0, 1\}^\ell$$
$$G(s) \in \{0, 1\}^L$$

## Deterministic

TRG  $G'$ 
$$R \in_r \{0, 1\}^L$$

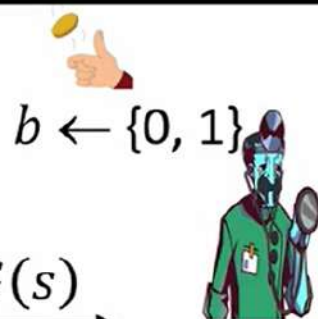
## Randomized

- ❑ The **output behavior** of  $G$  and  $G'$  should be **almost identical**
- ❑ No efficient algorithm (**distinguisher**) should be able to **distinguish apart** a random sample generated by  $G$ , from a random sample generated by  $G'$ , with a **significant probability**
- ❖ Modeled as an indistinguishability game

# PRG Indistinguishability Game

□  $b = 0 : y \in_r \{0, 1\}^L$

□  $b = 1:$



$b \leftarrow \{0, 1\}$

$y \in \{0, 1\}^L$   
(How  $y$  is generated?)



PPT Distinguisher  $\mathcal{D}$

$b'$

□ Algorithm  $G$  is a PRG, if for every PPT distinguisher  $\mathcal{D}$  participating in the above experiment:

$$\Pr(\mathcal{D} \text{ outputs } b' = b) \leq 1/2 + \text{negl}(n)$$

$\approx$

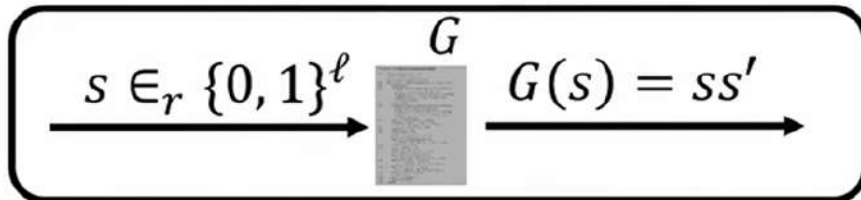
□ Algorithm  $G$  is a PRG, if for every PPT distinguisher  $\mathcal{D}$  participating in the above experiment:

$$\left| \underbrace{\Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=0]}_{\text{Prob. of } \mathcal{D} \text{ labeling } y \text{ as outcome of PRG, given that } y \text{ is generated by TRG}} - \underbrace{\Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=1]}_{\text{Prob. of } \mathcal{D} \text{ labeling } y \text{ as outcome of PRG, given that } y \text{ is generated by PRG}} \right| \leq \text{negl}(n)$$

Prob. of  $\mathcal{D}$  labeling  $y$  as outcome of PRG,  
given that  $y$  is generated by TRG

Prob. of  $\mathcal{D}$  labeling  $y$  as outcome of PRG,  
given that  $y$  is generated by PRG

# PRG : An Example



$$s' \stackrel{\text{def}}{=} s_1 \oplus s_2 \oplus \dots \oplus s_\ell$$

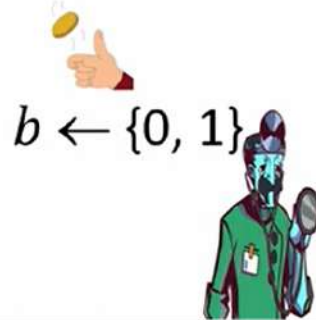
❑ Does there exist an **efficient statistical test** to distinguish a random sample of  $G$ , from a uniformly random  $(\ell + 1)$ -bit length string, with a significant probability ?

❖ For any  $y = (y_1, \dots, y_{\ell+1})$  where  $y = G(s)$ ,  $y_{\ell+1} = y_1 \oplus \dots \oplus y_\ell$  **always holds**

❖ For any  $y \in_r \{0, 1\}^{\ell+1}$ ,  $y_{\ell+1} = y_1 \oplus \dots \oplus y_\ell$  holds with probability  $\frac{1}{2}$

❑  $b = 0 : y \in_r \{0, 1\}^{\ell+1}$

❑  $b = 1 : y = G(s)$ , where  $s \in_r \{0, 1\}^\ell$



$b \leftarrow \{0, 1\}$

$y = (y_1, \dots, y_{\ell+1})$   
(How I generated it?)

$b' = 1$ , iff  $y_{\ell+1} = y_1 \oplus \dots \oplus y_\ell$

Distinguisher  $\mathcal{D}$



$$\Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=0] = \frac{1}{2}$$

$$\Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=1] = 1$$

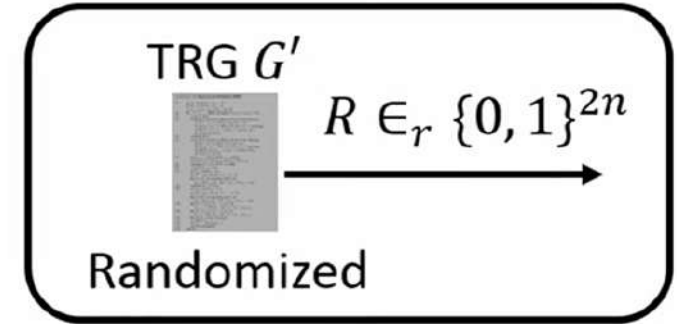
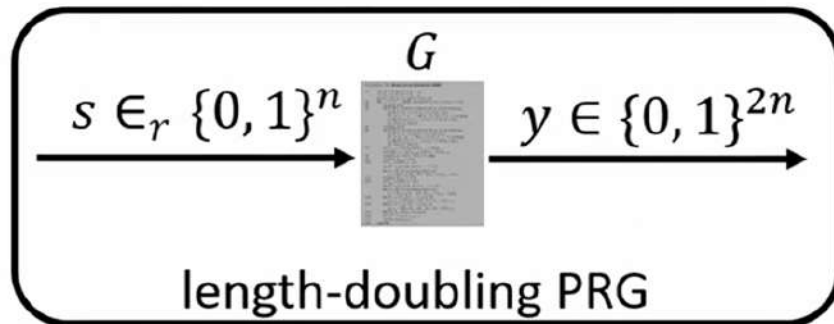
$$\begin{aligned} & \left| \Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=0] - \Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=1] \right| \\ &= \frac{1}{2} = \text{non-negl}(n) \end{aligned}$$



# PRG Can be Always Distinguished by a Brute force Distinguisher

❑ Any PRG has to **deterministically expand** its input

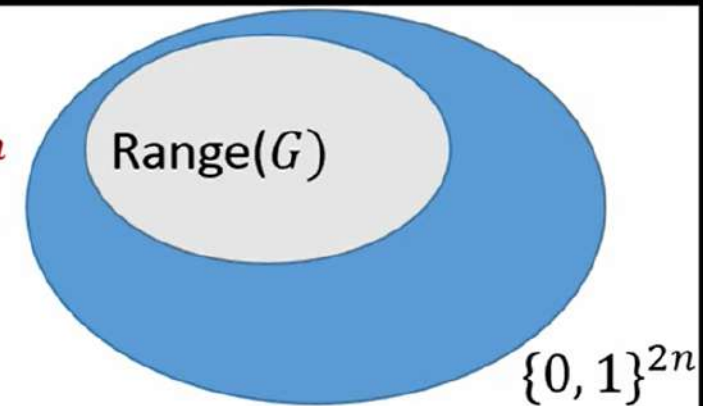
❖ Consequence : **output of PRG is far away from a uniformly random string**



❑ Most strings of length  $2n$  do not occur in the range of  $G$

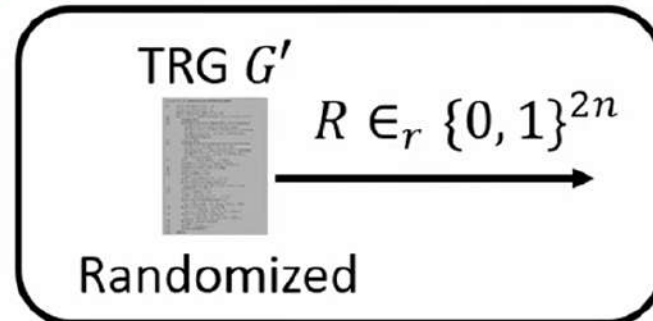
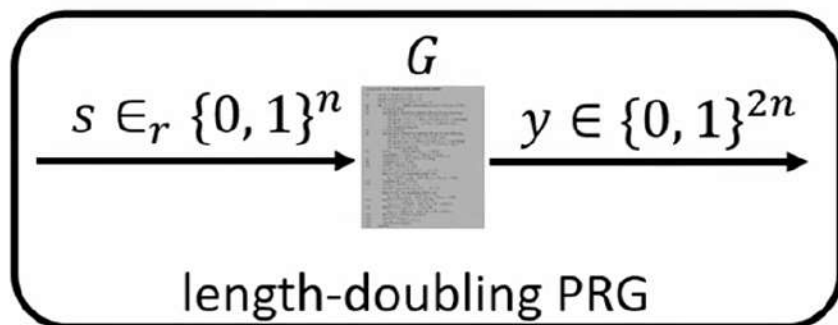
❖ Range of  $G$  --- **a proper subset of  $\{0, 1\}^{2n}$  of size at most  $2^n$**

❖ Prob. that a uniformly random  $2n$ -bit string occurs in the range of  $G$  is at most  $2^n / 2^{2n} = 2^{-n}$





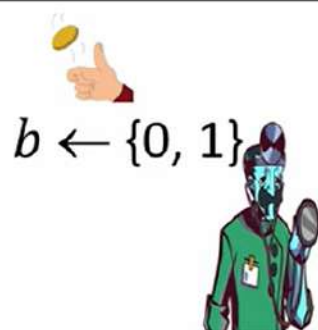
# PRG Can be Always Distinguished by a Brute force Distinguisher



□  $b = 0 : y \in_r \{0, 1\}^{2n}$

□  $b = 1: y = G(s)$ , where  $s \in_r \{0, 1\}^n$

□ Running time of  $\mathcal{D} = \mathcal{O}(2^n)$  --- inefficient



$y = (y_1, \dots, y_{2n})$   
(How  $y$  is generated?)

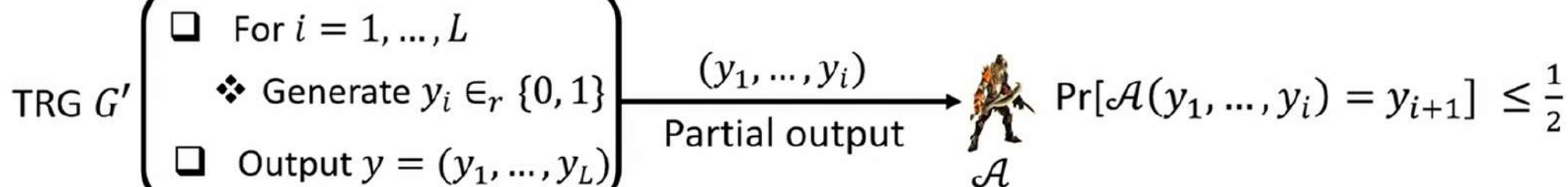
$b' = 1$ , iff  $y = G(s)$ , for some  $s \in \{0, 1\}^n$

Distinguisher  $\mathcal{D}$



$$| \Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=0] - \Pr[\mathcal{D} \text{ outputs } b'=1 \mid b=1] | = 1 - 2^{-n}$$

# PRG : An Alternate Definition



- ❖ For any  $i \in \{1, \dots, L-1\}$ , given the output bits  $y_1, \dots, y_i$  of  $y$ , no algorithm can predict the next output bit  $y_{i+1}$ , with probability better than  $\frac{1}{2}$
- ❖ PRG alternate definition (Next-bit predictor test) : the above should also hold for a PRG

