CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

January 22 2018

Review

- Perfect security
  - Impractical due to the requirements on key length
- Computational security
  - Relaxation of perfect security
  - PPT adversaries
  - Succeed with negligible probability
- EAV-secure encryption
  - Definition of security
  - Security game
  - Security experiment

Computational Security

Typically, we will say that a **scheme C is secure** if

$\forall$ **Probabilistic polynomial-time** algorithm A

Pr[A(n) "breaks the scheme" C(n)] is **negligible** in n.

- Scheme **C** and the **adversary A** take input **security parameter**.
- 2 relaxations of perfect security
  - PPT adversary
  - Adversary can succeed, but with very small probability (negligible)

Perfect vs. Computational Security

we will require that $\mathbf{m_0,m_1}$ are chosen by a **poly-time adversary**

Recall: An encryption scheme is **perfectly secret** if for all $\mathbf{m_0,m_1, c}$

$$\Pr[\mathrm{Enc}(K, m_0) = c] = \Pr[\mathrm{Enc}(K, m_1) = c]$$

Meaning: no attacker can distinguish $\mathbf{Enc(K, m_0)}$ from $\mathbf{Enc(K, m_1)}$

New: no PPT attacker can distinguish $\mathbf{Enc(K, m_0)}$ from $\mathbf{Enc(K, m_1)}$ with better then negligible probability.

Security Game

Π= (Gen,Enc,Dec): an encryption scheme

security parameter **n**

PPT Adversary A

Alice Challenger

chooses m_0, m_1 such that $|m_0| = |m_1|$

→ m_0, m_1 →

1. Choose $k \leftarrow Gen(n)$
2. chooses random $b \leftarrow \{0,1\}$
3. calculate $c \leftarrow Enc(k, m_b)$

← c ←

Makes a guess **b'**

Security definition:
We say that **(Gen,Enc,Dec)** is **indistinguishable against eavesdropping (EAV-secure)** if for any **polynomial time** adversary, **$\Pr[b'=b] - ½$** is negligible in n.

The security definition

- Experiment $\mathrm{Exp}_{\Pi,A}^{\mathrm{EAV}}(n)$:
  1. Choose $k \leftarrow Gen(n)$
  2. $m_0, m_1 \leftarrow A_1\ (n)$
  3. $b \leftarrow^R \{0,1\}; c \leftarrow Enc_k(m_b)$
  4. $b' \leftarrow A_2\ (m_0, m_1, c)$
  5. Output 1 if $b = b'$ and 0 otherwise

We say that **(Gen, Enc,Dec)** is **EAV-secure** (secure against eavesdropping) if:

For every **PPT** adversary $A = (A_1, A_2)$:

|Pr[$\mathrm{Exp}_{\Pi,A}^{\mathrm{EAV}}(n)$ = 1]- ½ | negligible in n

Construct secure encryption

- Impossible to construct from scratch

Suppose that **G** is a "pseudorandom generator"

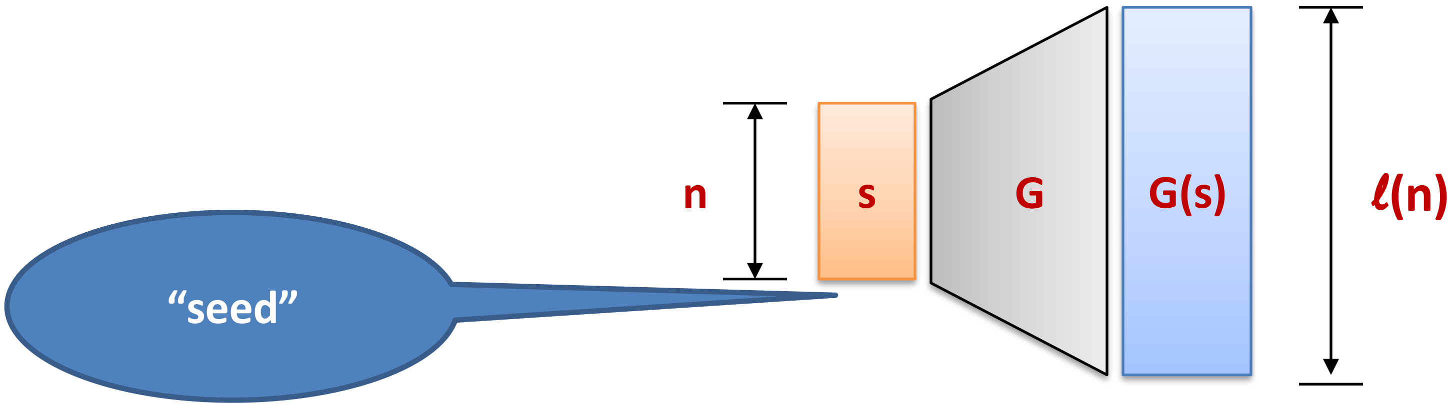
↓

We can construct a computationally secure encryption scheme based on **G**

Outline

- Pseudorandom generators
  - Security definition
  - Examples
  - Proofs by reduction
- PRG implies EAV-secure encryption
  - Using PRG to shorten key in one-time pad
  - Reduction proof

Pseudorandom generator: G



A pseudorandom generator is a deterministic algorithm
$G : \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$.

- **Output length:** $\ell(n)$ for all s with $|s| = n$ we have $|G(s)| = \ell(n)$.
- **Stretch:** $\ell(n) - n$

Goal (imprecise): If s chosen randomly from $\{0,1\}^n$,
then $G(s)$ "looks" like it was chosen randomly from $\{0,1\}^{\ell(n)}$.

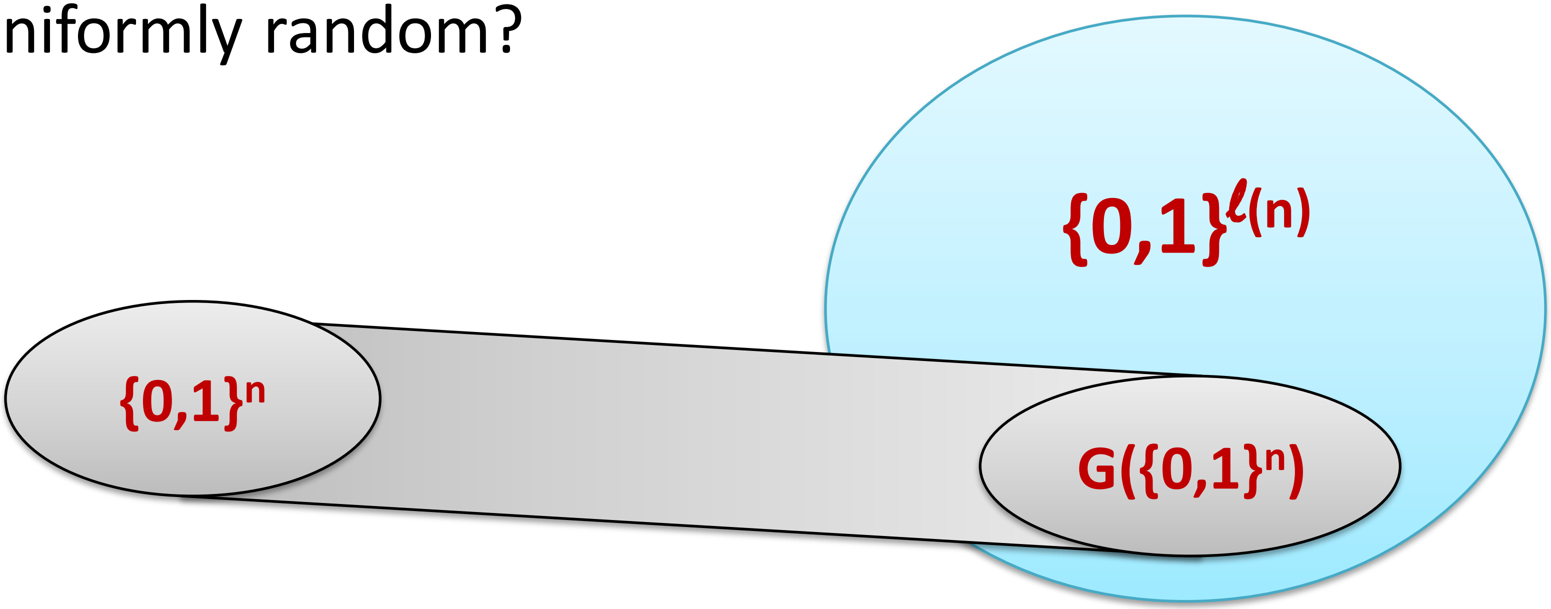
"Looks random"

Suppose $s \in \{0,1\}^n$ is chosen randomly.

Can

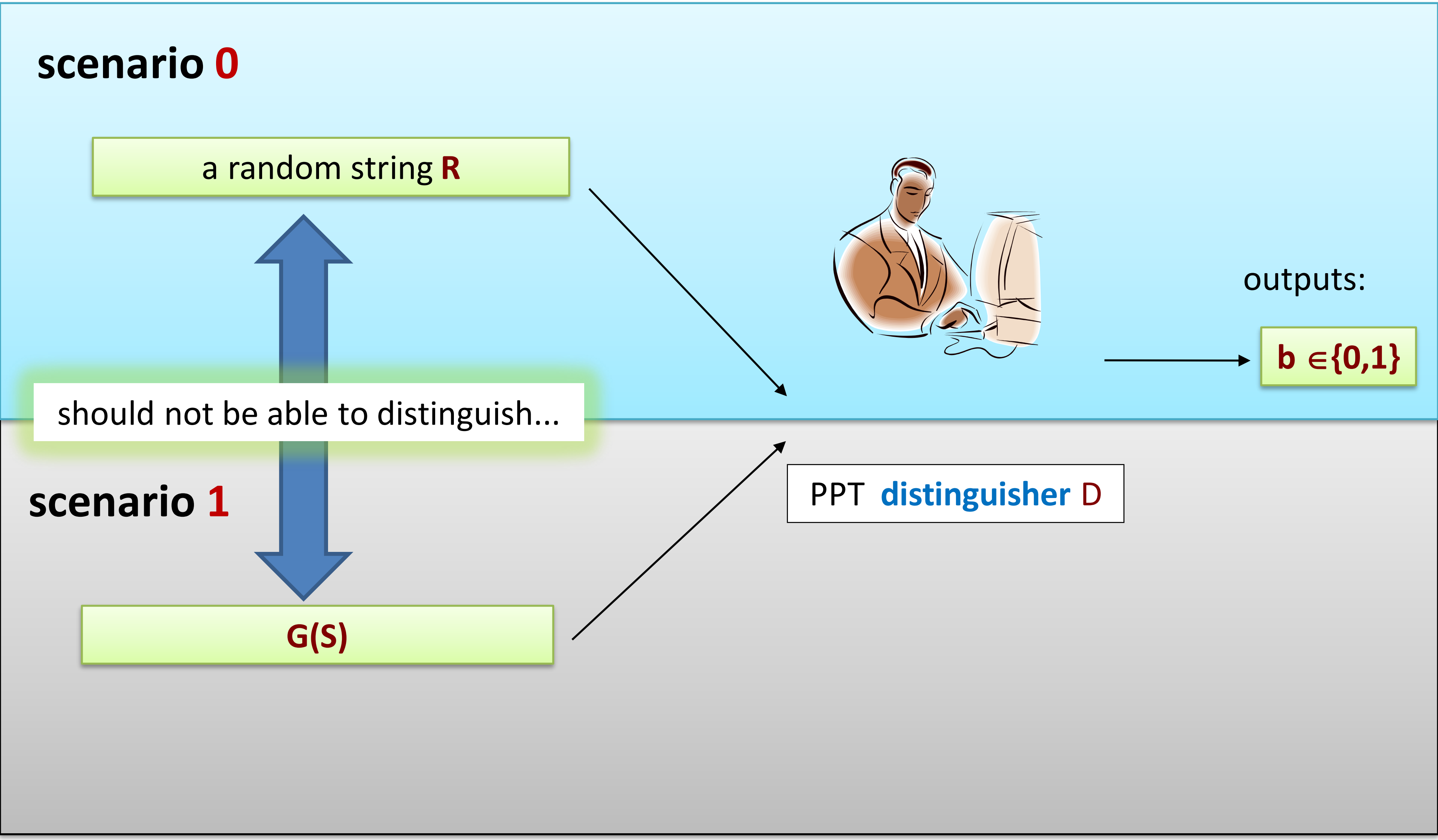
$$G(s) \in \{0,1\}^{\ell(n)}$$

be uniformly random?



Computationally indistinguishable

PRG – main idea of the definition



Cryptographic PRG

outputs:

a random string **r**

or

G(s) (where **s** random) →

0 if he thinks it's **r**

1 if he thinks it's **G(s)**

Should not be able to distinguish...

Definition

n – a parameter

s – a variable distributed uniformly over $\{0,1\}^n$

r – a variable distributed uniformly over $\{0,1\}^{\ell(n)}$

Definition: **G** is a **secure** **PRG** if for every PPT algorithm **D** we have:

$$| \Pr[D(G(s)) = 1] - \Pr[D(r) = 1] |$$

is negligible in **n**.

PRG Example 1

- Define $G: \{0,1\}^n \to \{0,1\}^{n+1}$ as:

  $G(s_1 \cdots s_n) = s_1 \cdots s_n s_{n+1}$,where $s_{n+1} = s_1 \oplus \cdots \oplus s_n$
- Is G a secure PRG?

Build distinguisher D for G; D is given string u

D outputs 1 if $\mathrm{u}_{\mathrm{n+1}} = u_1 \oplus \cdots \oplus u_n$

- World 0 - u = r random: $\Pr[D(r) = 1] = \frac{1}{2}$
- World 1 - u = G(s): $\Pr[D(G(s)) = 1] = 1$

  $|\Pr[D(r) = 1]$ - $\Pr[D(G(s)) = 1]|$ = ½

PRG Example 2

- Assume $G: \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ is a PRG
- Define $G': \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ as:

  $$G'(s) = \bar{G}(s) = G(s) \oplus 1^{\ell(n)}$$

- Is G’ a secure PRG?

G secure PRG → G’ secure PRG

Reduction proof

Distinguisher D’ for G’ →

Distinguisher D for G

PRG Example 2

Assume $G: \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ is a PRG

Define $G': \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ as: $G'(s) = \bar{G}(s)$

- Let D’ be a distinguisher for G’ with prob $\epsilon(n)$ non-negligible

  $|\Pr[D'(r) = 1]$ - $\Pr[D'(G'(s)) = 1]$= ϵ(n)

- Design D dist. for G
  - D given string u ($u = \mathrm{G}(\mathrm{s})$ in world 1 and $u = \mathrm{r}$ random in world 0)
  - D gives $\bar{u}$ input to D’ and outputs what D’ outputs
- World 0: $\Pr[D(r) = 1] = \Pr[D'(r) = 1]$
- World 1: $\Pr\left[D\left(G(s)\right) = 1\right] = \Pr\left[D'\left(\bar{G}(s)\right) = 1\right]$

Thus:

$$| \Pr[D(r) = 1] - \Pr\left[D\left(G(s)\right) = 1\right] |$$

$$= \quad |\Pr[D'(r) = 1] - \Pr[D'(G'(s)) = 1]|$$

$$= \epsilon(n)$$

PRG Example 3

- Assume $G_1, G_2: \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$ are PRGs
- Define $G: \{0,1\}^n \rightarrow \{0,1\}^{2\ell(n)}$ as:

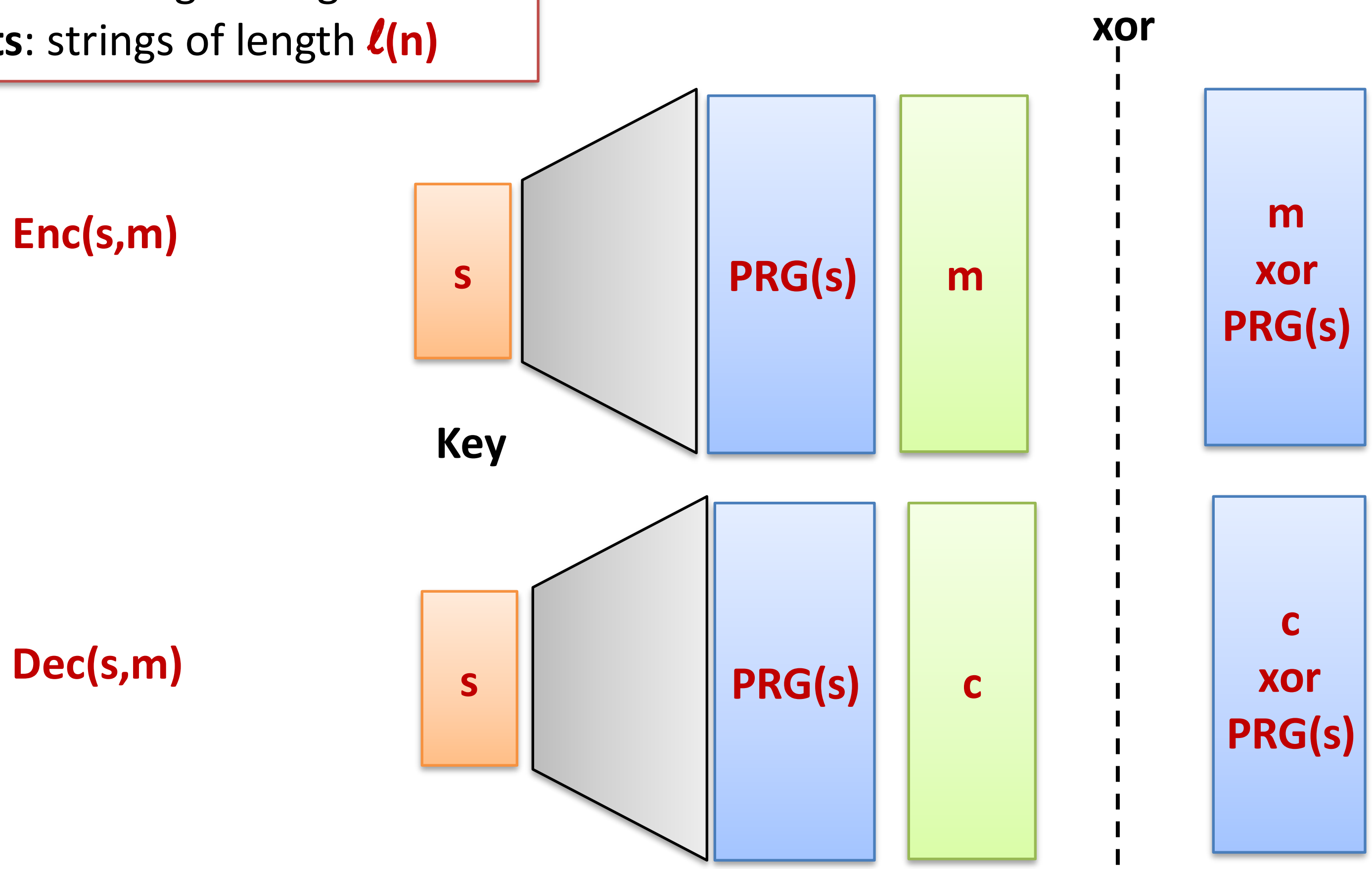
  $G(s) = G_1(s)||G_2(s)$
- Is G a secure PRG?
- Take $G_2(s) = \bar{G}_1(s), \text{then } G(s) = G_1(s)\bar{G}_1(s)$
- Build D distinguisher for G; D given string $u = u_1 u_2$
- D outputs 1 if $\mathrm{u}_2 = \bar{u}_1$
- World 0 - u = r random: $\Pr[D(r) = 1] = \frac{1}{2^{\ell(n)}}$
- World 1 - u = G(s): $\Pr[D(G(s)) = 1] = 1$

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| = 1 - \frac{1}{2^{\ell(n)}}$$

Using a PRG to build efficient OTP

Use PRGs to “shorten” the key in the one time pad

Key: random string of length **n**
Plaintexts: strings of length ***ℓ*(n)**

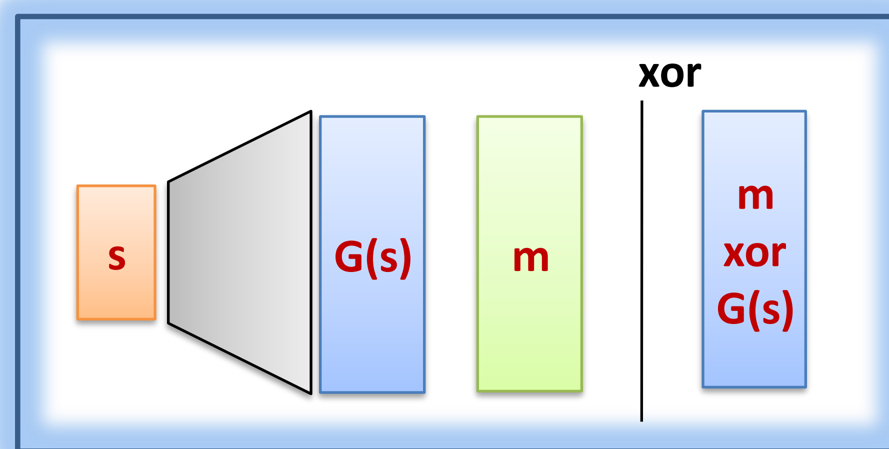


EAV-secure one-time pad

Theorem

(for simplicity consider only the single message case)

If **G** is a **secure PRG** then the **encryption** scheme constructed before is *secure*.

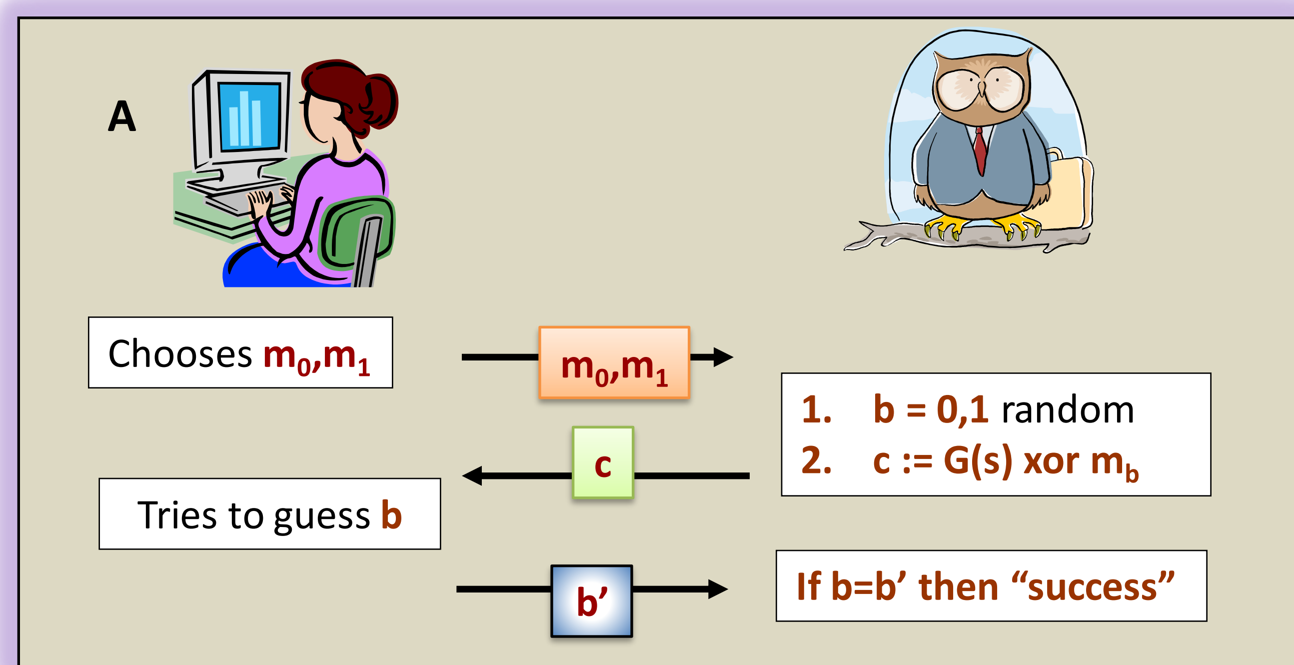


cryptographic PRGs exist → EAV-secure encryption exists

Reduction proof

Attack on encryption → Attack on PRG

Recall: Security Game



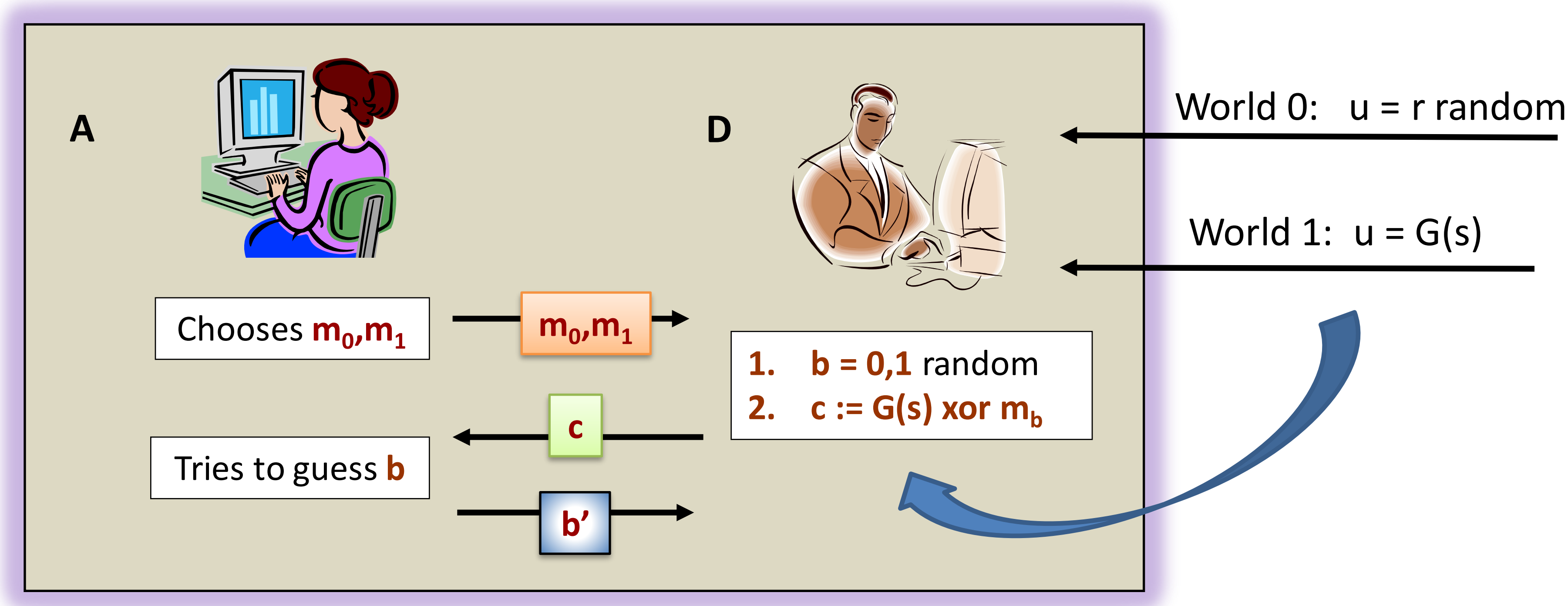
If exists PPT "encryption attacker" A that breaks security of encryption:

Pr["guess b correctly"] = $\frac{1}{2} + \delta(n)$.

where δ is not negligible.

Then exists PPT "PRG distinguisher" that break security of PRG **G**.

Design distinguisher D for PRG



Let A be PPT attacker that breaks security of encryption:

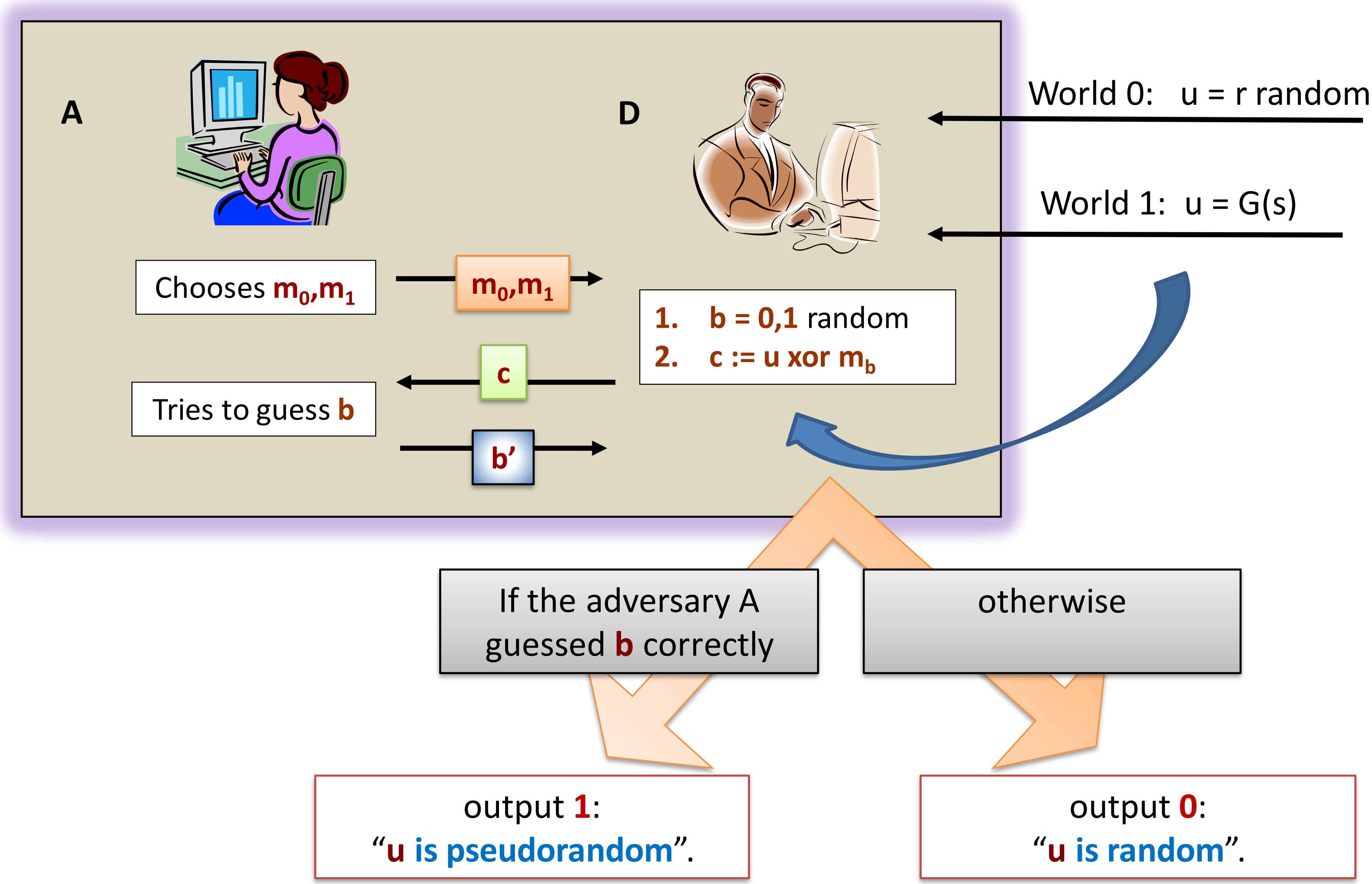
Pr[b' =b] = $\frac{1}{2} + \delta(n)$ where δ is not negligible.

Design PPT "PRG distinguisher" D that breaks security of PRG **G**.

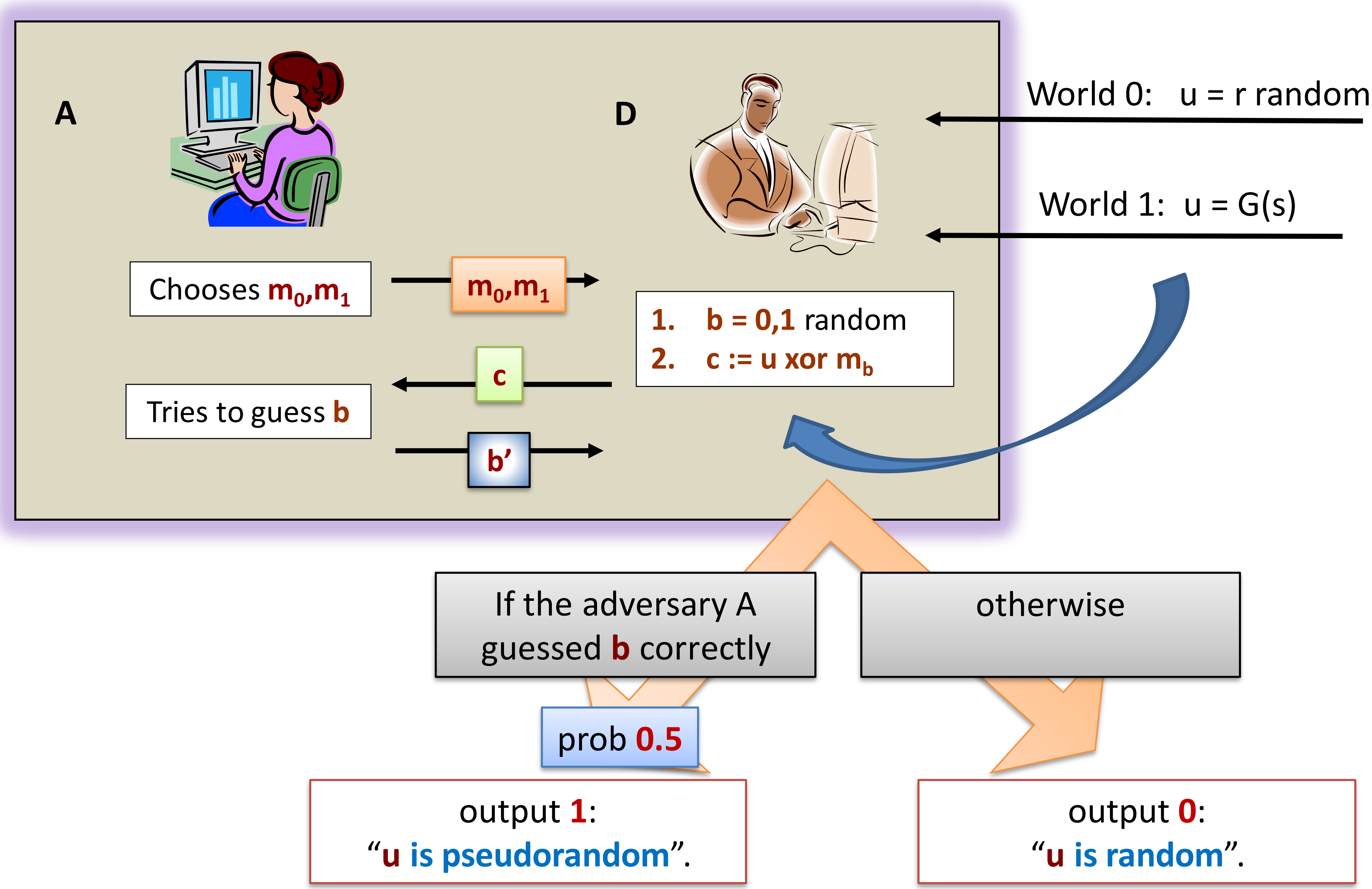
D is given an input u (either random string or G(s)) and needs to distinguish them.

D interacts with A by playing the challenger

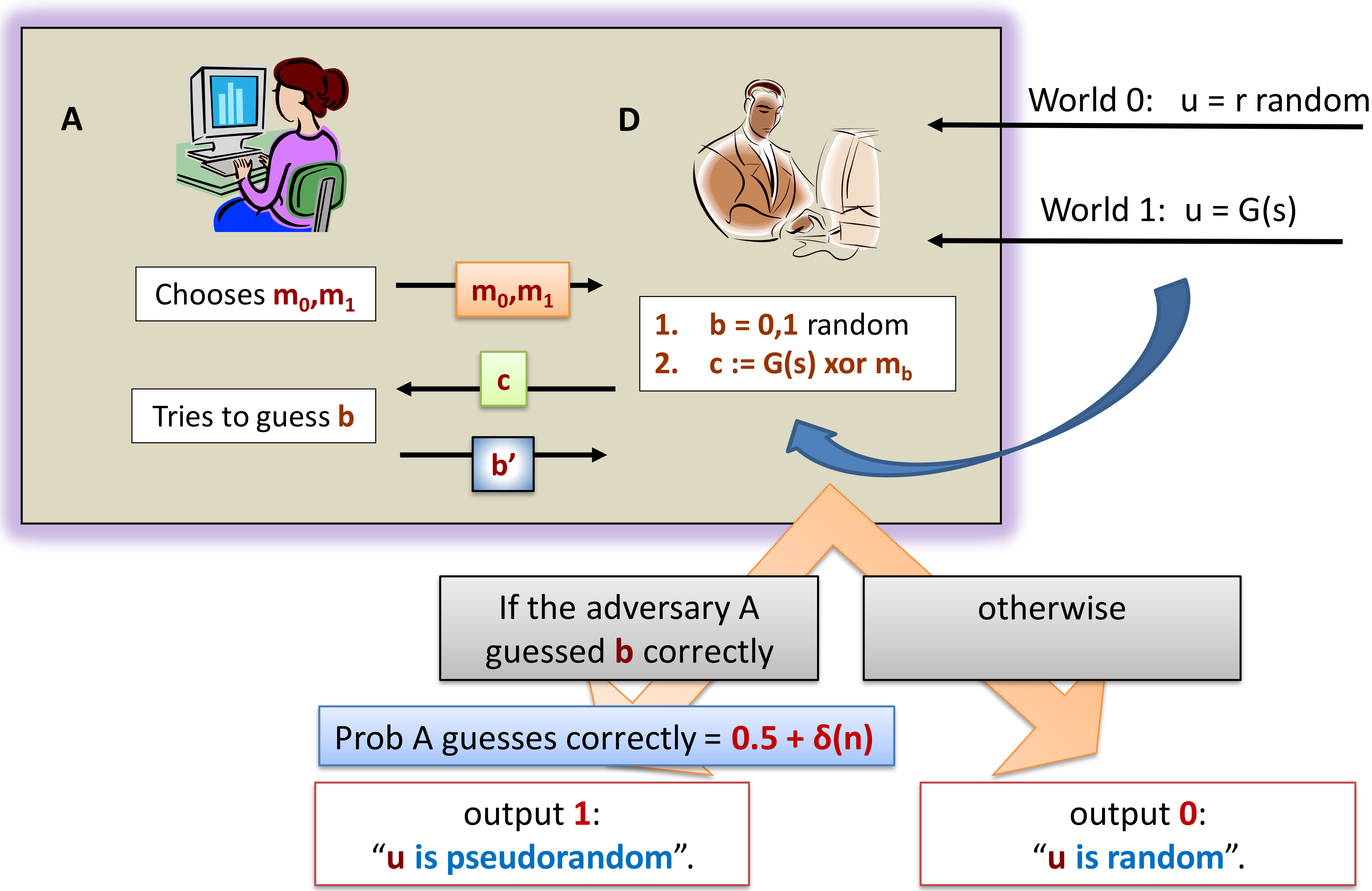
Design distinguisher D for PRG



"World 0": u is a random string



“World 1”: x = G(S)


A
D
World 0: u = r random
World 1: u = G(s)
Chooses m_0,m_1
m_0,m_1
1. b = 0,1 random
2. c := G(s) xor m_b
c
Tries to guess b
b’
If the adversary A guessed b correctly
otherwise
Prob A guesses correctly = 0.5 + δ(n)
output 1: “u is pseudorandom”.
output 0: “u is random”.

Hence

u is a random string **r**

u = G(s)

the adversary **A** guesses **b** correctly with probability **0.5**

the adversary A guesses **b** correctly with probability **0.5 + δ(n)**

outputs: **Pr[D(r) = 1] = .5**

Pr [D(G(s)) = 1] = .5+δ(n)

$$\left| P(D(r) = 1) - P(D(G(s)) = 1) \right| = \left| 0.5 - (0.5 + \delta(n)) \right| = \delta(n)$$

Distinguisher **D** breaks the PRG!

The complexity

The distinguisher  simply simulated

one execution of the adversary 

Hence he works in polynomial time.

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/