Introduction to Modern Cryptography

3rd lecture:

Computational Security of Private-Key Encryption and Pseudorandomness

some of these slides are copied from or heavily inspired by the
University College London MSc InfoSec 2010 course given by Jens Groth
Thank you very much!

last time:

- perfectly secure encryption
- one-time pad
- its limitations

3rd lecture (today):

- computational security
- pseudorandomness
- reduction proof

Turing Machine

- Simple well-defined mathematical model of computation
- Church-Turing Thesis:
  Turing machines can compute anything that is computable (they are *universal*).
- Measure time in steps a Turing machine takes (think of a step as a clock-cycle on processor)
- Number of steps is "robust", it is related to time in other more realistic models

Efficiency

- Definition:
  An efficient Turing machine is one that runs in time t(*n*) polynomial in the input length *n*.
- Natural examples:
  - $t(n) = n^2$ is efficient
  - $t(n) = 2^n$ is not efficient
- Not so natural examples:
  - $t(n) = n^{100} + 1000000000000$ is efficient
  - $t(n) = 2^{n-1000000}$ is not efficient

Polynomial time

- Why define efficient as polynomial time?
- Combining two poly-time Turing machines gives poly-time Turing machine:
  - poly(n) + poly(n) = poly(n)
  - poly(n) poly(n) = poly(n)
  - poly(poly(n)) = poly(n)
- At least better than exponential time
- Experience shows that security against poly-time adversary corresponds well with real life security

Probabilistic Turing Machines

- May make random choices. We model this by giving it additional randomness $r \leftarrow \{0,1\}^*$.
- We write $y \leftarrow \mathrm{Adv}(x)$ or $y := \mathrm{Adv}(x;r)$ when adversary runs on input x with randomness r
- PPT: probabilistic polynomial-time
- players and adversaries are modeled as PPT Turing machines

Negligible Advantage

- We want the adversary's advantage $\varepsilon(n)$ to decrease as we increase the security parameter
- Definition:
  We say a function $\varepsilon(n)$ is ***negligible*** if for all polynomials poly(n) we have

$$\varepsilon(n) < 1 / \mathrm{poly}(n)$$

  for all sufficiently large n.

Negligible Advantage

- We say a function $\varepsilon(n)$ is negligible if for all polynomials poly(n) we have

  $$\varepsilon(n) < 1 / \mathrm{poly}(n)$$

  for all sufficiently large n.
- Natural examples:
  - 2^{-n} is negligible
  - n^{-1} is not negligible
- Less natural examples:
  - $2^{1000000-n}$ is negligible
  - n^{-100} is not negligible
- Closed under composition:
  - $\mathrm{negl}(n) + \mathrm{negl}(n) = \mathrm{negl}(n)$
- Resists polynomial scaling:
  - $\mathrm{poly}(n)\ \mathrm{negl}(n) = \mathrm{negl}(n)$

Negligible Advantage

Intuition: events occurring with negligible probability are so unlikely that they can be ignored for all practical purposes.

- Natural examples:
  - 2^{-n} is negligible
  - n^{-1} is not negligible
- Less natural examples:
  - $2^{1000000-n}$ is negligible
  - n^{-100} is not negligible
- Closed under composition:
  - $\mathrm{negl}(n) + \mathrm{negl}(n) = \mathrm{negl}(n)$
- Resists polynomial scaling:
  - $\mathrm{poly}(n)\ \mathrm{negl}(n) = \mathrm{negl}(n)$

Definition 3.7

A private-key encryption scheme is a tuple of PPT algorithms (Gen,Enc,Dec) such that:

1. The key-generation algorithm Gen takes as input the security parameter n and outputs a key k: $k \leftarrow \mathrm{Gen}(1^n)$. Assume: $|k| \geq n$.
2. for a plaintext message $m \in \{0,1\}^*$ ciphertext $c \leftarrow \mathrm{Enc}_k(m)$
3. for ciphertext c, we have $m := \mathrm{Dec}_k(c)$.

Correctness: For every n, every k output by $\mathrm{Gen}(1^n)$,every m, it holds that $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$.

Definition 3.7

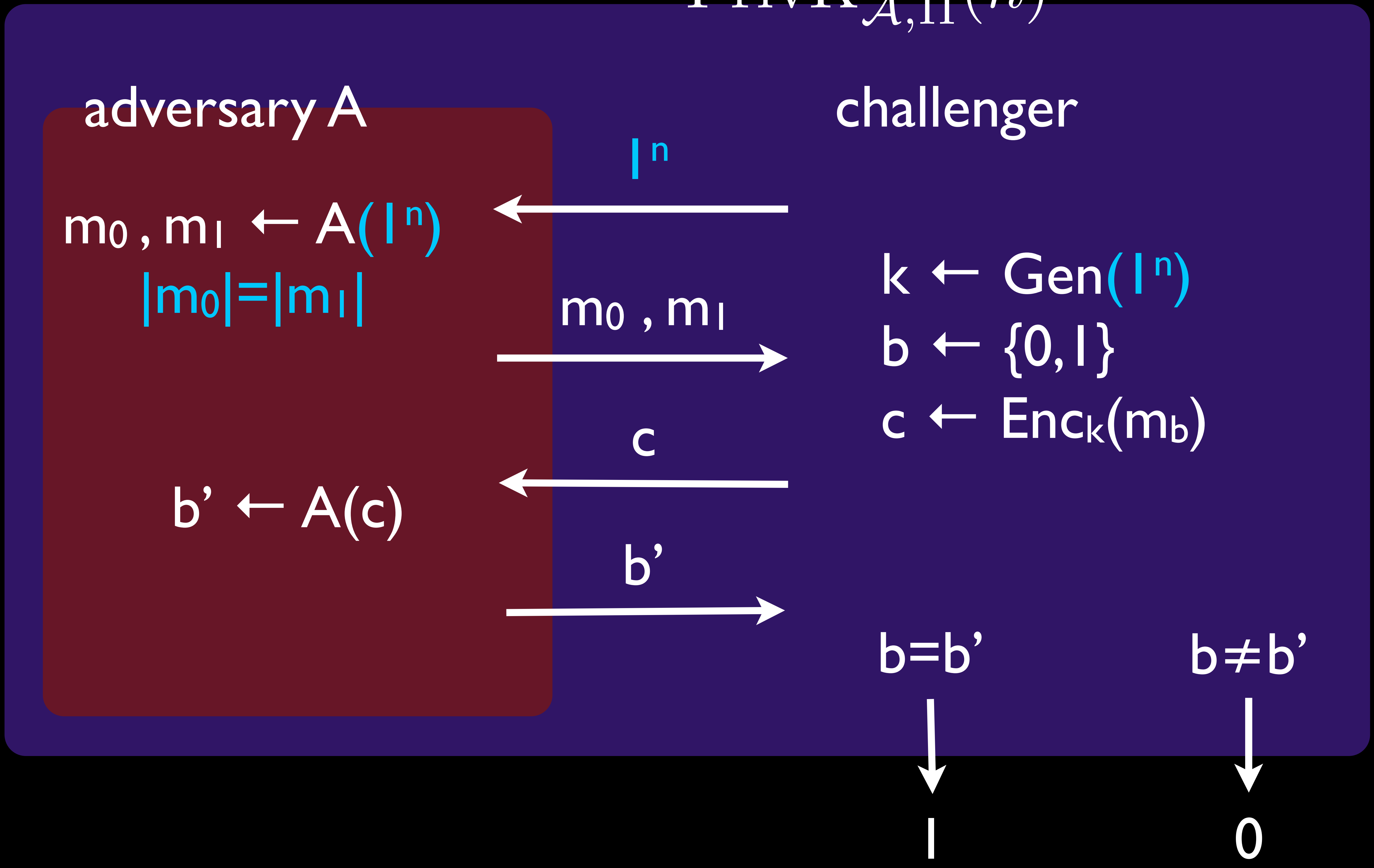
A fixed-length private-key encryption scheme is a tuple of PPT algorithms (Gen,Enc,Dec) such that:

1. The key-generation algorithm Gen takes as input the security parameter n and outputs a key k: $k \leftarrow \mathrm{Gen}(1^n)$. Assume: $|k| \geq n$.
2. for a plaintext message $m \in \{0,1\}^{\ell(n)}$ ciphertext $c \leftarrow \mathrm{Enc}_k(m)$
3. for ciphertext c, we have $m := \mathrm{Dec}_k(c)$.

Correctness: For every n, every k output by $\mathrm{Gen}(1^n)$,every m, it holds that $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$.

Eavesdropping Indistinguishability Experiment

$$\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$$



adversary A

challenger

1^n

$m_0, m_1 \leftarrow A(1^n)$

$|m_0|=|m_1|$

m_0, m_1

$k \leftarrow \mathrm{Gen}(1^n)$

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

$c \leftarrow \mathrm{Enc}_k(m_b)$

c

$b' \leftarrow A(c)$

b’

$b=b'$ → 1

$b\neq b'$ → 0

Silvio Micali

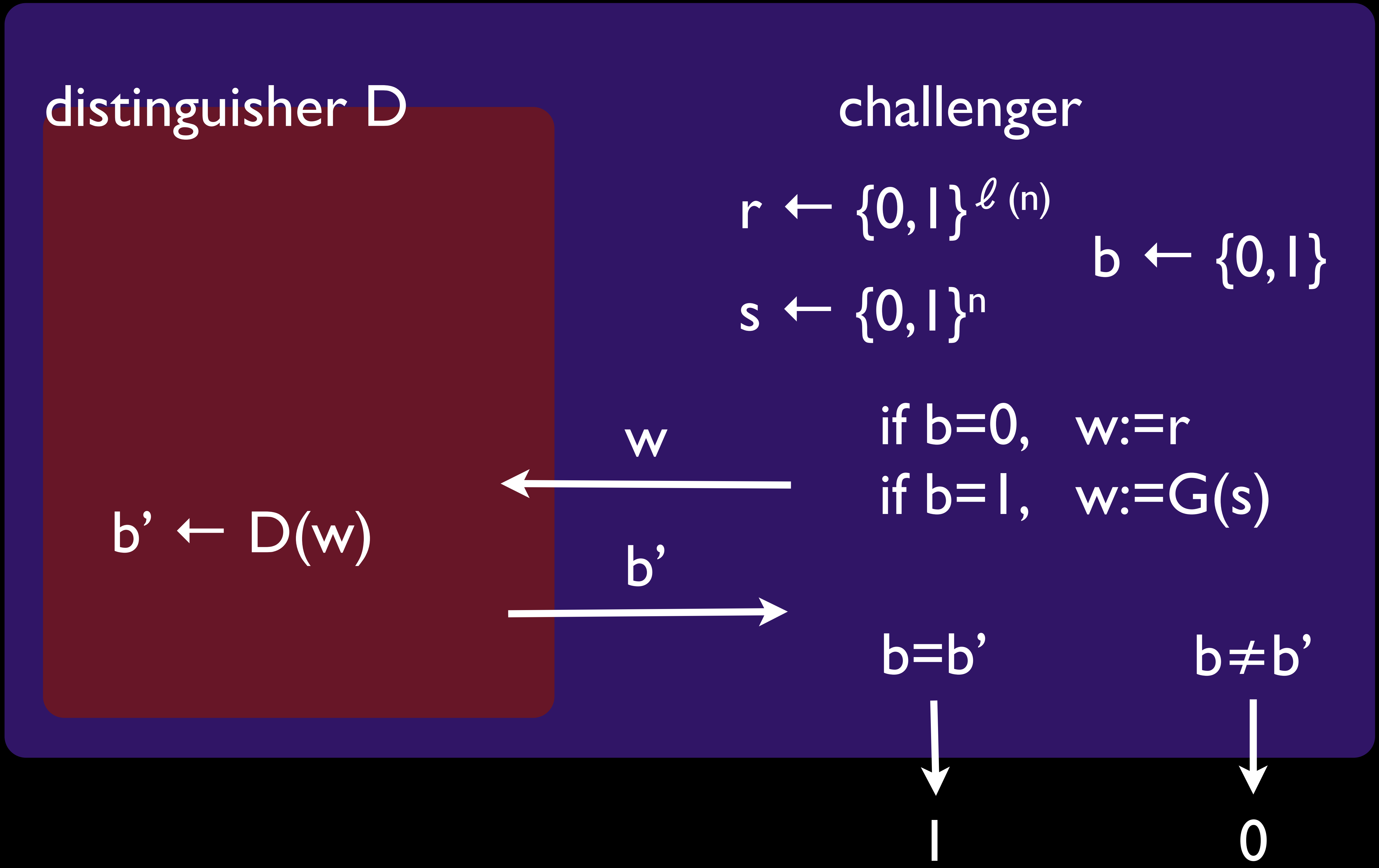
Shafi Goldwasser

1984:

- semantic security
- indistinguishability

PRG Indistinguishability Experiment

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



distinguisher D

challenger

$r \leftarrow \{0,1\}^{\ell(n)}$

$s \leftarrow \{0,1\}^n$

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

if $b=0$, $w:=r$

if $b=1$, $w:=G(s)$

w

$b' \leftarrow D(w)$

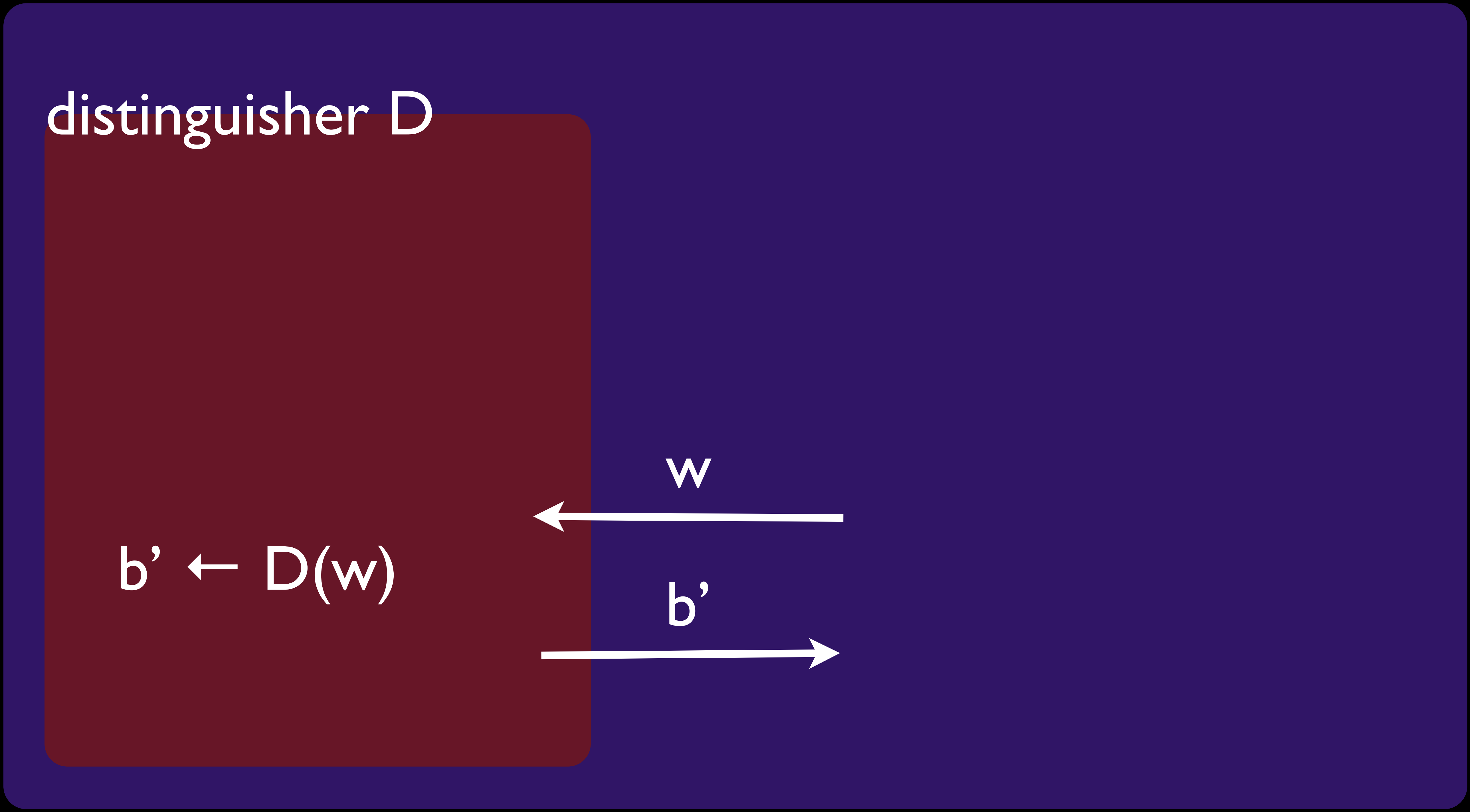
b'

$b=b'$ $b\neq b'$

1 0

PRG Indistinguishability Experiment

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



distinguisher D

$b' \leftarrow D(w)$

w

b'

Silvio Micali

Manuel Blum

- 1984: defined notion of pseudo-random generator

Andrew Chi-Chih Yao

- PhD from Stanford and Chicago
- Tsinghua University in Beijing
- definition of PRGs and constructions

- winner of Knuth prize and Turing Award