

## CS409m: Introduction to Cryptography

Lecture 04 (08/Aug/25)

Instructor: Chethan Kamath

# Annonouncements

- Lab Exercise 1 (graded) will be out today (08/Aug)
  - Will be discussed in TA session today
- You should have registered on <https://cs409m.ctfd.io/>
  - See Moodle announcement by Nilabha for instructions
- Assignment 2 (ungraded) will also be out today (08/Aug)
- No lectures next week :( /.)
  - Open house on 13 Aug and Ind. Day on 15/Aug

# Recall from Previous Lecture

- Task: secure communication with shared keys
- Threat model: perfect secrecy against eavesdroppers



## Attack Model: Eavesdropping

- 1 Eve computationally unbounded (deterministic)
- 2 Knows description of  $\Pi$  (Kerchhoff's principle)
- 3 Shared key is hidden from Eve
- 4 Can eavesdrop and learn ciphertext



## Break Model: Perfect secrecy

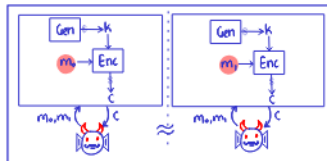
- Eve learns *no information* about the message

- 1 Shannon's definition

$$\Pr[M = m^* | C = c^*] = \Pr[M = m^*]$$

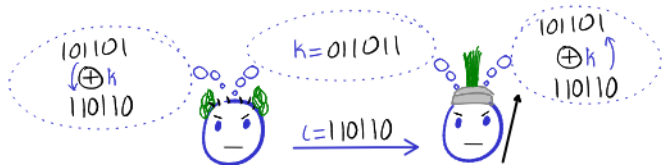


- 2 Two worlds definition



## Recall from Previous Lecture...

- Task: secure communication with shared keys
- Threat model: **perfect secrecy** against **eavesdroppers**



+ **Theorem 1** (Shannon): one-time pad (OTP) is perfectly secret

— Limitations of OTP:

- Keys are **as large as** messages  $|\mathcal{K}| = |\mathcal{M}|$

⚠ Becomes **insecure** if key re-used: see Lab Exercise 1


— **Theorem 2** (Shannon): For **any** perfectly-secret SKE,  $|\mathcal{K}| \geq |\mathcal{M}|$

# Plan for Today's Lecture

- Bypass Shannon's barrier by *relaxing* the **threat model**:
  - “Impossible to break” → “Infeasible to break”
  - Computational secrecy against eavesdroppers




## Attack Model: Eavesdropping

- 1 Eve is *efficient*: PPT 
- 2 Knows description of  $\Pi$  (Kerchhoff's principle)
- 3 Shared key hidden from Eve
- 4 Can eavesdrop and learn ciphertext



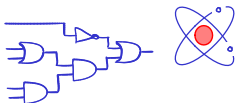
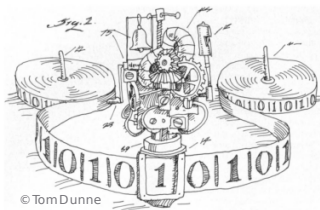
## Break Model: Secrecy, w.h.p.

- Eve may learn information, but only with “low probability” 
- Negligible probability

★ Both relaxations are necessary!

# Plan for Today's Lecture...

## Models of Computation



## Negligible Functions



© Joy Shrader/Wikipedia

# Plan for Today's Lecture...

- 1 Models of Computation: A Primer
- 2 Negligible Functions
- 3 Computational Secrecy Against Eavesdroppers

# Plan for Today's Lecture

- 1 Models of Computation: A Primer
- 2 Negligible Functions
- 3 Computational Secrecy Against Eavesdroppers

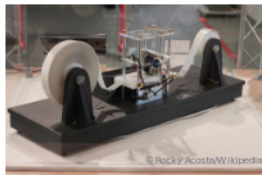


# Models of Computation: Turing Machine

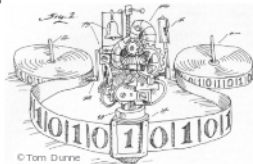
- We have **informally** introduced (randomised) algorithms
  - Set of instructions or rules that carries out a computation
- To **formally** study algorithms, we need a *model of computation*
  - How to define running time?
  - What does “efficient” mean?
  - ...
- E.g.: **Turing machine** (TM)
  - Introduced by Turing as “automatic machine”
  - Mathematically precise model of computation
- Components:
  - Tapes: to provide input, for memory...
  - States: “halt” “good so far”
  - Transition function/rule: “processor”



© Rocky Acosta/Wikipedia

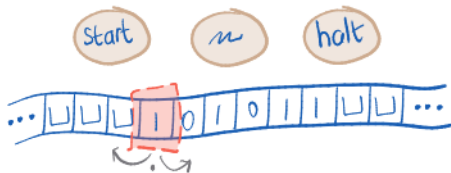


© Rocky Acosta/Wikipedia



© Tom Durne

# How Turing Machines Work

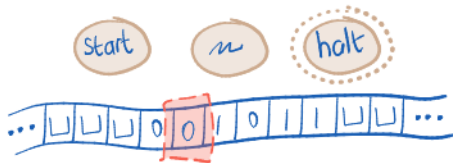


## Definition 1 ([AB09], §1.2)

A  $k$ -tape Turing Machine  $M$  is described by a tuple  $(\Gamma, Q, \tau)$  such that:

- $M$  has  $k$  memory tapes (input/work/output tapes) with **heads**
- $\Gamma$  is a finite alphabet, which includes a special "blank" symbol  $\sqcup$ 
  - Every memory cell has an element from  $\Gamma$
- $Q$  is a finite set of **states**
  - Special states: "start" and "halt/done"
- $\tau$  is a function from  $Q \times \Gamma^k$  to  $Q \times \Gamma^k \times \{\rightarrow, \leftarrow, \cdot\}^k$ 
  - Transition function/rule: encodes behaviour of  $M$

# How Turing Machines Work...



## ■ Start configuration

- The tape is initialised with the input string
- Rest of the tape is blank (□)
- The head is at the start of the input
- State is start

## ■ Computation step

- Apply  $\tau$  on current state and input to obtain next state, output and next head position



## ■ Halting

- Stop computation if state is halt



Demo: [turingmachine.io](http://turingmachine.io)

# Running Time of Turing Machine

## Definition 2

Let  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  and  $T : \mathbb{N} \rightarrow \mathbb{N}$  be functions, and  $M$  be a TM

- $M$  *computes*  $f$  if on every input  $x \in \{0, 1\}^*$  (placed in input tape),  $M$  halts with  $f(x)$  on its output tape
- $M$  *computes*  $f$  *in time*  $T$ , if for every input  $x \in \{0, 1\}^*$ ,  $M$  halts as above within  $T(|x|)$  steps

❓ What was the running time of the TM in the demo?

- **Efficient** (deterministic) computation:  $T$  is any *fixed polynomial*
  - E.g.,  $T(n) := n^3$  or  $T(n) = 4n^{1000} + \log(n)$
- **Efficient** *randomised* computation \$\$\$
  - Also referred to *probabilistic polynomial time* (PPT) ★
  - Definition 2 extended to *randomised* TM

## Other Models of Computation Exist



❓ Is your laptop a Turing Machine? Not quite, closer to

- RAM Machine

- TM with fixed-sized tape



Can move head to *any position* in the work tape in *one step*



❓ What about a basic calculator? Closer to

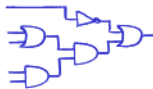
- Boolean circuit (family)

- Represented using gates (AND, OR, NOT) and wires

- One circuit for each input length

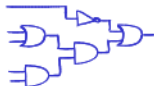
- Size of the circuit is the *number of its gates*

- *Efficient circuits*: size is polynomial (in input length)



# Compromise I: We Restrict Eve to PPT TM

- Why TM? Church-Turing thesis:
  - “Every physically realizable computation device – whether it’s based on silicon, DNA, neurons, or some other alien technology – can be *simulated (efficiently)* by a Turing machine.”\* ([AB09])
  - To rephrase: the exact model of computation **doesn’t matter**
- Why PPT? “Captures” efficient computation
  - Real-world adversaries assumed to be efficient
  - Polynomials have nice closure properties
  - Randomness allowed since it is allowed for honest algorithms
- Some stronger models for Eve:
  - Polynomial-sized family of circuits: allows “non-uniform” advice
  - Quantum polynomial-time algorithms



---

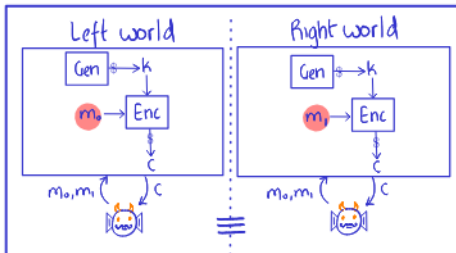
\*Possible exceptions: Boolean circuit family, quantum TM

# First Attempt at Computational Secrecy

## Candidate Definition 1 (Computational Secrecy)


An SKE  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is **computationally-secret** if for every PPT eavesdropper *Eve*

$$\Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_0)}}} [\text{Eve}(c) = 0] - \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_1)}}} [\text{Eve}(c) = 0] = 0$$



# First Attempt at Computational Secrecy


## Candidate Definition 1 (Computational Secrecy)


An SKE  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is  **computationally-secret** if for every PPT eavesdropper *Eve*

$$\Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_0)}}} [\text{Eve}(c) = 0] - \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_1)}}} [\text{Eve}(c) = 0] = 0$$


## Exercise 1

Show that Shannon's impossibility extends to Candidate Definition 1

 Hint 1: use similar approach as in proof of Theorem 2 (Lecture 03)

 Hint 2: exploit randomness for efficiency



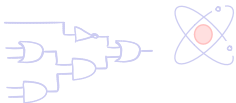
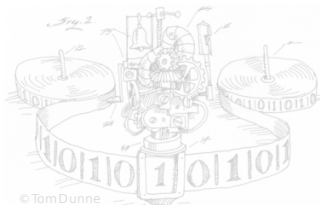
 Take-away: even *Eve* can distinguish with “very low” probability



# Plan for Today's Lecture...



## Models of Computation



## Negligible Functions



© Joy Shrader/Wikipedia

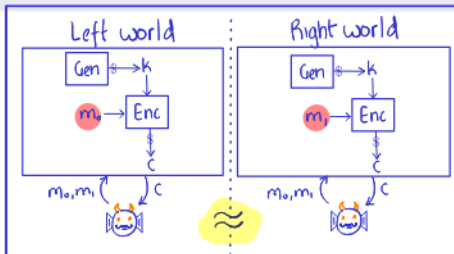
## Compromise II: Eve Learns with Low Probability

- Eve may learn information, but only with “low probability”  $\delta$
- ❓ How to quantify “low probability”? First attempt:  $\delta \approx 1/|\mathcal{K}|$

### Candidate Definition 2 (Computational Secrecy)

An SKE  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is **computationally-secret** against **eavesdroppers** if for every *PPT Eve*

$$\left| \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_0)}}} [\text{Eve}(c) = 0] - \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_1)}}} [\text{Eve}(c) = 0] \right| = \delta$$



## Compromise II: Eve Learns with Low Probability

- Eve may learn information, but only with “low probability”  $\delta$
- ❓ How to quantify “low probability”? First attempt:  $\delta \approx 1/|\mathcal{K}|$

### Candidate Definition 2 (Computational Secrecy)

An SKE  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is **computationally-secret** against **eavesdroppers** if for every *PPT Eve*

$$\left| \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_0)}}} [\text{Eve}(c) = 0] - \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \text{Gen} \\ c \leftarrow \text{Enc}(k, m_1)}}} [\text{Eve}(c) = 0] \right| = \delta$$

### Exercise 2

- Does Shannon's impossibility extend also to Candidate Definition 2?  
💡 Hint: Work out precise probability of learning in Exercise 1
- Can Eve trivially succeed with  $1/|\mathcal{K}|$  probability? (💡 Hint: guess the key?)

★ Take-away:  $1/|\mathcal{K}|$  too low

## Second Compromise: Break is a Low-Probability Event

### Definition 3

A function  $f : \mathbb{N} \rightarrow \mathbb{R}^+$  is *negligible* if for every polynomial  $p$  and sufficiently large  $n$ ,  $f(n) < 1/p(n)$  holds.

❓ Negligible or not?

1  $f_1(n) := 1/314159n^{314159}$

2  $f_2(n) := 1/2^n$

## Second Compromise: Break is a Low-Probability Event

### Definition 3

A function  $f : \mathbb{N} \rightarrow \mathbb{R}^+$  is *negligible* if for every polynomial  $p$  and sufficiently large  $n$ ,  $f(n) < 1/p(n)$  holds.

❓ Negligible or not?

👎 1  $f_1(n) := 1/314159n^{314159}$

👍 2  $f_2(n) := 1/2^n$  → "inverse exponential"

👎 3  $f_3(n) := \begin{cases} 1/2^n & \text{for odd } n \\ 1/314159n^{314159} & \text{for even } n \end{cases}$  "hybrid of  $f_1$  &  $f_2$ "

👍 4  $f_4(n) := n^{-\log(n)}$  → "inverse quasi-polynomial"

👍 5  $f_5(n) := p(n)/2^n$ , for a very large polynomial  $p(n)$

👍 6  $f_6(n) := n^{\log(n)}/2^n$

★ To show that  $f(n)$  is *non-negligible*, show that there exists a polynomial  $p$  such that  $f(n) > 1/p(n)$  for *infinitely often*  $ns$ .

# Plan for Today's Lecture

- 1 Models of Computation: A Primer
- 2 Negligible Functions
- 3 Computational Secrecy Against Eavesdroppers

# The Security Parameter



- So far towards defining computational secrecy
  - Restrict to **PPT Eve**
  - Allow **negligible probability** of Eve learning
- When designing scheme, want ability to precisely control above values via a parameter  $n$  (sometimes denoted by  $\lambda$ ):
  - Want: Honest algorithms run in time **fixed polynomial** in  $n$
  - Allow: **Eve** can run in time **arbitrary polynomial** in  $n$
  - Require: **Eve** to have a success probability **negligible** in  $n$
- $n$  is the “security parameter”
  - Determines amount of time (generally resources) required to “break” scheme

# How to Choose the Right Security Parameter?...



- Suppose a cryptography designer claims that Eve running in  $n^3$  mins can break his scheme with probability  $2^{40}/2^n$

❓ What  $n$  do you choose while implementing?



- 1  $n \leq 40$ ? Eve working for  $40^3$  mins  $\approx$  6 weeks can break with probability 1

- Not very safe! ⚠️



- 2  $n = 50$ ? Eve working for  $50^3$  mins  $\approx$  3 months can break with probability  $1/1000$

- May be acceptable



- 3  $n = 500$ ? Eve working for  $500^3$  mins  $\approx$  200 years can break with probability  $2^{-460}$

- Quite safe ★

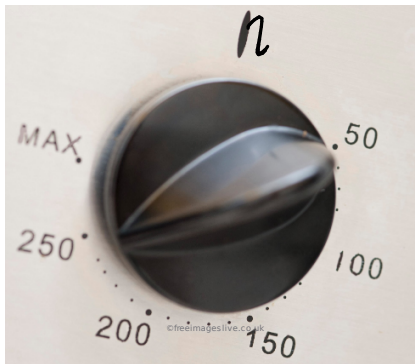


# How to Choose the Right Security Parameter?...

- Why not set  $n$  to be very high to be very safe?

less efficient!

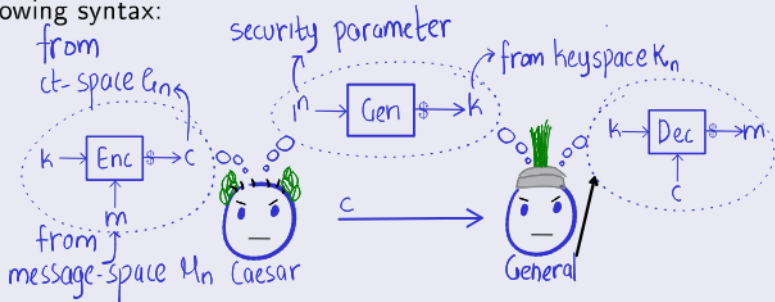
more secure



# Incorporating Security Parameter into SKE Definition

## Definition 4 (Shared/Symmetric-Key Encryption (SKE))

An SKE  $\Pi$  is a triple of efficient algorithms (Gen, Enc, Dec) with the following syntax:



- Correctness of decryption: for every  $n \in \mathbb{N}$ , message  $m \in M_n$ ,

$$\Pr_{k \leftarrow \text{Gen}(1^n), c \leftarrow \text{Enc}(k, m)} [\text{Dec}(k, c) = m] = 1$$

# Let's Finally Define Computational Secrecy!

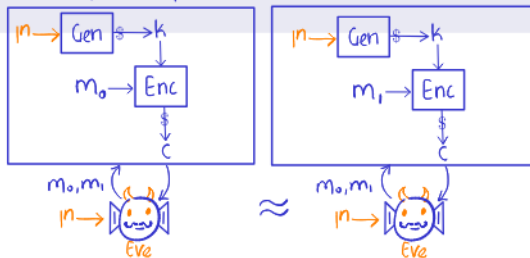
## Definition 5 (Two-Worlds Definition)

An SKE  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is **computationally-secret** against **eavesdroppers** if for every PPT Eve

$$\delta(n) := \left| \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve}(1^n) \\ k \leftarrow \text{Gen}(1^n) \\ c \leftarrow \text{Enc}(k, m_0)}}} [\text{Eve}(c) = 0] - \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve}(1^n) \\ k \leftarrow \text{Gen}(1^n) \\ c \leftarrow \text{Enc}(k, m_1)}}} [\text{Eve}(c) = 0] \right|$$

0 world                      1 world

is negligible.



# Let's Finally Define Computational Secrecy!

## Definition 5 (Two-Worlds Definition)

An SKE  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  is **computationally-secret** against **eavesdroppers** if for every **PPT Eve**

$$\delta(n) := \left| \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve}(1^n) \\ k \leftarrow \text{Gen}(1^n) \\ c \leftarrow \text{Enc}(k, m_0)}}} [\text{Eve}(c) = 0] - \Pr_{\substack{(m_0, m_1) \leftarrow \text{Eve}(1^n) \\ k \leftarrow \text{Gen}(1^n) \\ c \leftarrow \text{Enc}(k, m_1)}}} [\text{Eve}(c) = 0] \right|$$

is negligible.

## Exercise 3

Does Definition 5 change if we quantify for all pair of messages  $(m_0, m_1)$  instead of adversarially choosing it?

# More Generally: Computational Indistinguishability

## Definition 6 (computational indistinguishability)

Two distributions  $X_0$  and  $X_1$  are *computationally indistinguishable* if for every **PPT** distinguisher  $D$ ,

$$\delta(n) := \Pr_{x \leftarrow X_0} [D(x) = 0] - \Pr_{x \leftarrow X_1} [D(x) = 0]$$

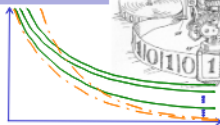
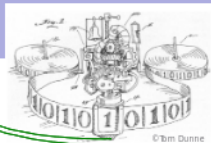
is negligible.

- Computational secrecy against eavesdroppers can be rephrased as: the ciphertext distribution in the left and the right worlds are computationally indistinguishable.

## Exercise 4

Formally show the above

# Recap/Next Lecture

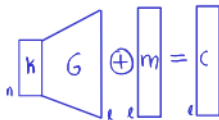


## ■ To recap:

- Introduced Turing Machines and PPT
- Introduced negligible functions
- Established the notion of computational secrecy against eavesdroppers by relaxing the threat model
  - **Attack model**: restrict to PPT Eves **NEW**
  - **Break model**: allow break with negligible probability **NEW**
- Defined **computational indistinguishability**: we'll use this notion throughout the course

## ■ Next lecture:

- Pseudorandom generators (PRG)
- Computationally-secret SKE scheme: "Computational OTP"
- First security reduction!



❓ More Questions? ❓

## References

- 1 §3.1 in [KL14] for more details on computational secrecy
- 2 Chapter 1 in [AB09] for more about Turing machines. The original paper is [Tur37]
- 3 [turingmachine.io](http://turingmachine.io) for visualisation of Turing machines



Sanjeev Arora and Boaz Barak.

*Computational Complexity - A Modern Approach.*

Cambridge University Press, 2009.



Jonathan Katz and Yehuda Lindell.

*Introduction to Modern Cryptography (3rd ed.).*

Chapman and Hall/CRC, 2014.



Alan M. Turing.

On computable numbers, with an application to the entscheidungsproblem.

*Proc. London Math. Soc.*, s2-42(1):230–265, 1937.