

---

CSE 3400 - Introduction to Computer & Network Security  
(aka: Introduction to Cybersecurity)

Lecture 2  
Encryption – Part I

Ghada Almashaqbeh

UConn

From Textbook Slides by Prof. Amir Herzberg

UConn

---

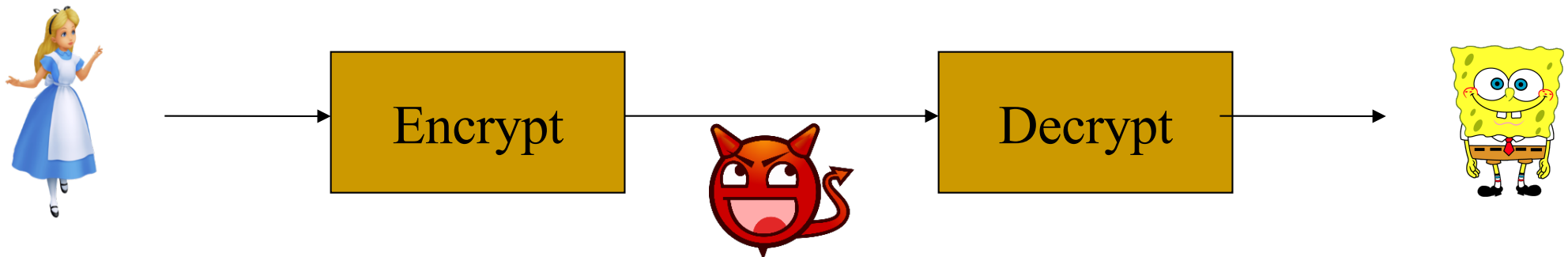
---

# Outline

- Introduction and motivation.
  - Ancient ciphers.
  - Kerckhoffs' Principle.
  - Encryption attacker models.
-

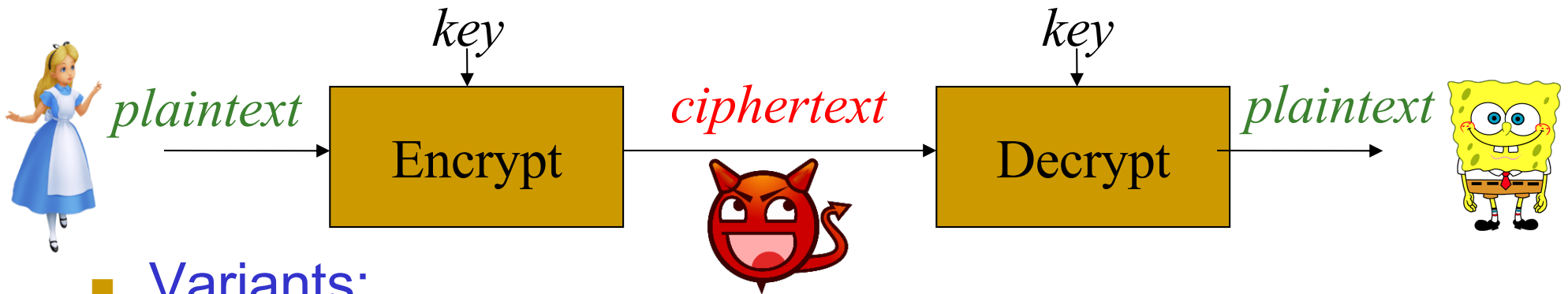
# Encryption

- Prevention of exposure of secret information
- Earliest and `basic' tool of cryptology
- Related terms:
  - Cryptography: `secret writing'
  - Cryptanalysis: `breaking' encryption
  - Encryption scheme = Cryptosystem = Cipher



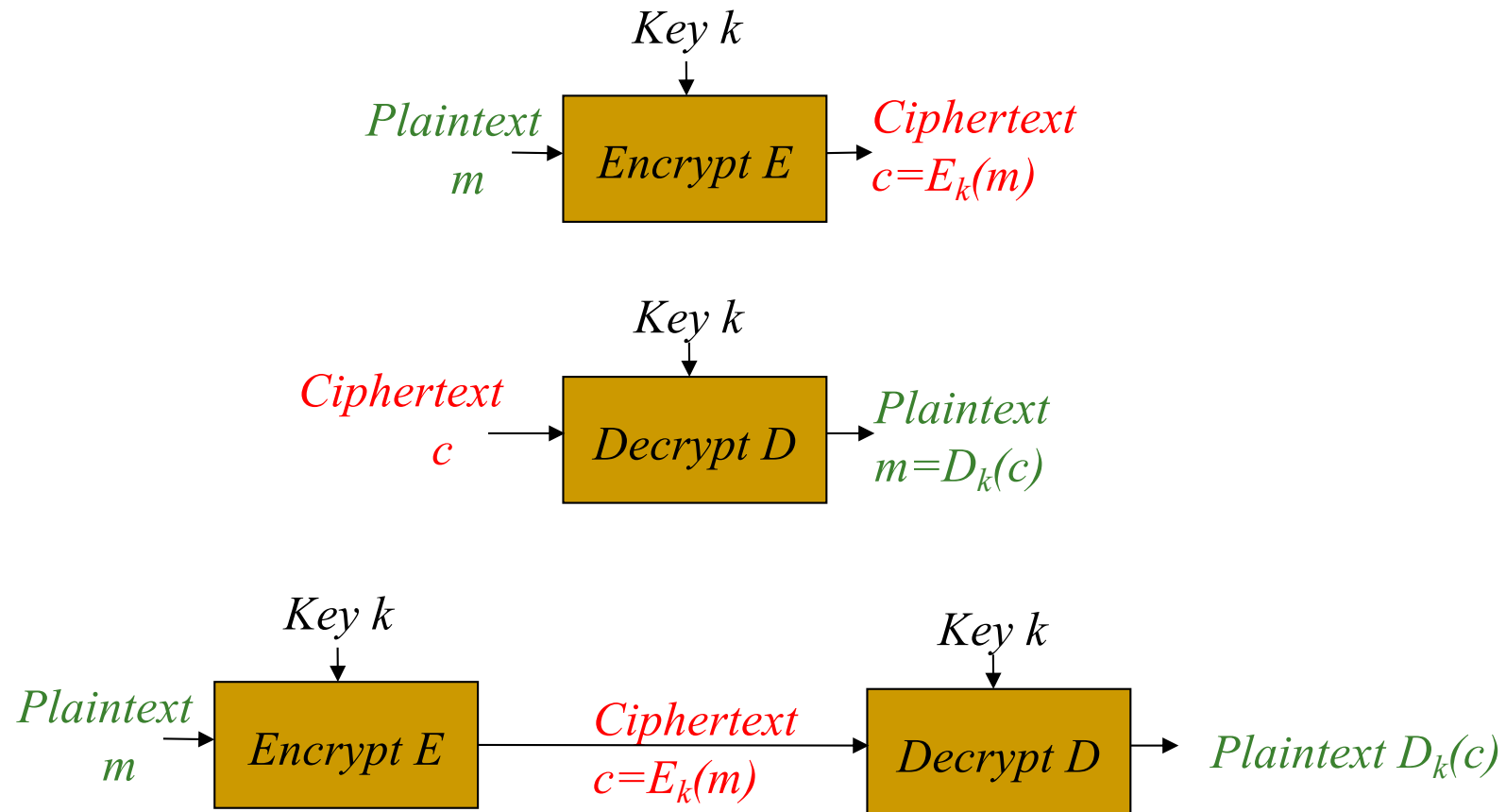
# The Encryption World: basic terms

- Goal: **encrypt** plaintext into ciphertext
- Only legit-recipient can **decrypt** ciphertext to plaintext
  - Adversary cannot learn anything from ciphertext
- Idea: decryption uses secret algorithm and/or key



- Variants:
  - Keyed or unkeyed?
  - Shared key (symmetric) or public/private keys (asymmetric)?
  - Stateful / stateless ? Randomized ? Input size ?

# Encryption, Decryption, Correctness



**Correctness:**  $m = D_k(E_k(m))$

---

# Ancient, Keyless Ciphers

- Ancient ciphers were simple, naive
  - No key: secrecy is in the algorithm
- Monoalphabetic ciphers: encrypt/decrypt one character at a time
  - Plaintext, ciphertext are both single letters
  - A set  $\{<E,D>\}$  of permutation + inverse:  $m=D(E(m))$

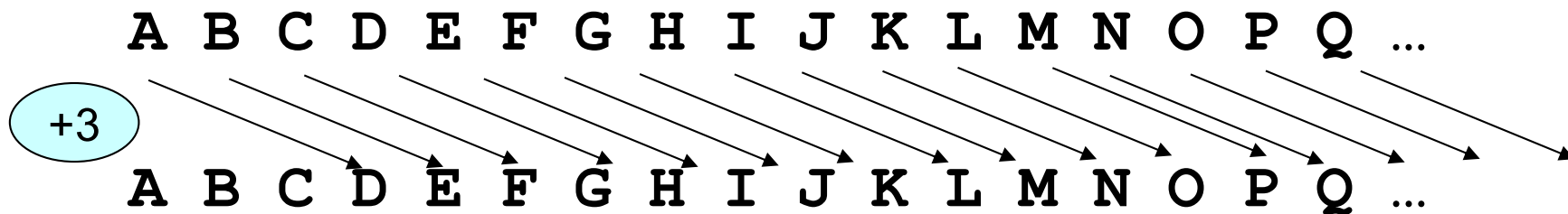
# AzBy Cipher

- AzBy Cipher
  - Based on the (Hebrew) At-BaSh, the first documented cipher [Jeremiah]
  - Substitute the first letter of alphabet by the last... and so on:
- Mathematically: Let A be 0, B be 1, ..., Z be 25. Let m denote plaintext and c denote ciphertext.
  - $c = \text{Enc}(m) = 25 - m$
  - $m = \text{Dec}(c) = 25 - c$

Plaintext:	A	B	C	.	.	.	.	X	Y	Z
	↓	↓	↓					↓	↓	↓
Ciphertext:	Z	Y	X	.	.	.	.	C	B	A

# (Unkeyed) Caesar Cipher

- Used by Julius Caesar
- **Rotate** the 26 letters of the alphabet by 3:



- As formula:

$$c = E(m) = m + 3 \pmod{26}$$

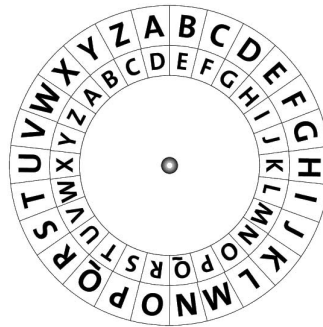
$$m = D(c) = c - 3 \pmod{26}$$

- Caesar and AzBy are trivial to cryptanalyze
  - No key – algorithm itself is `secret`
  - `Security by obscurity`



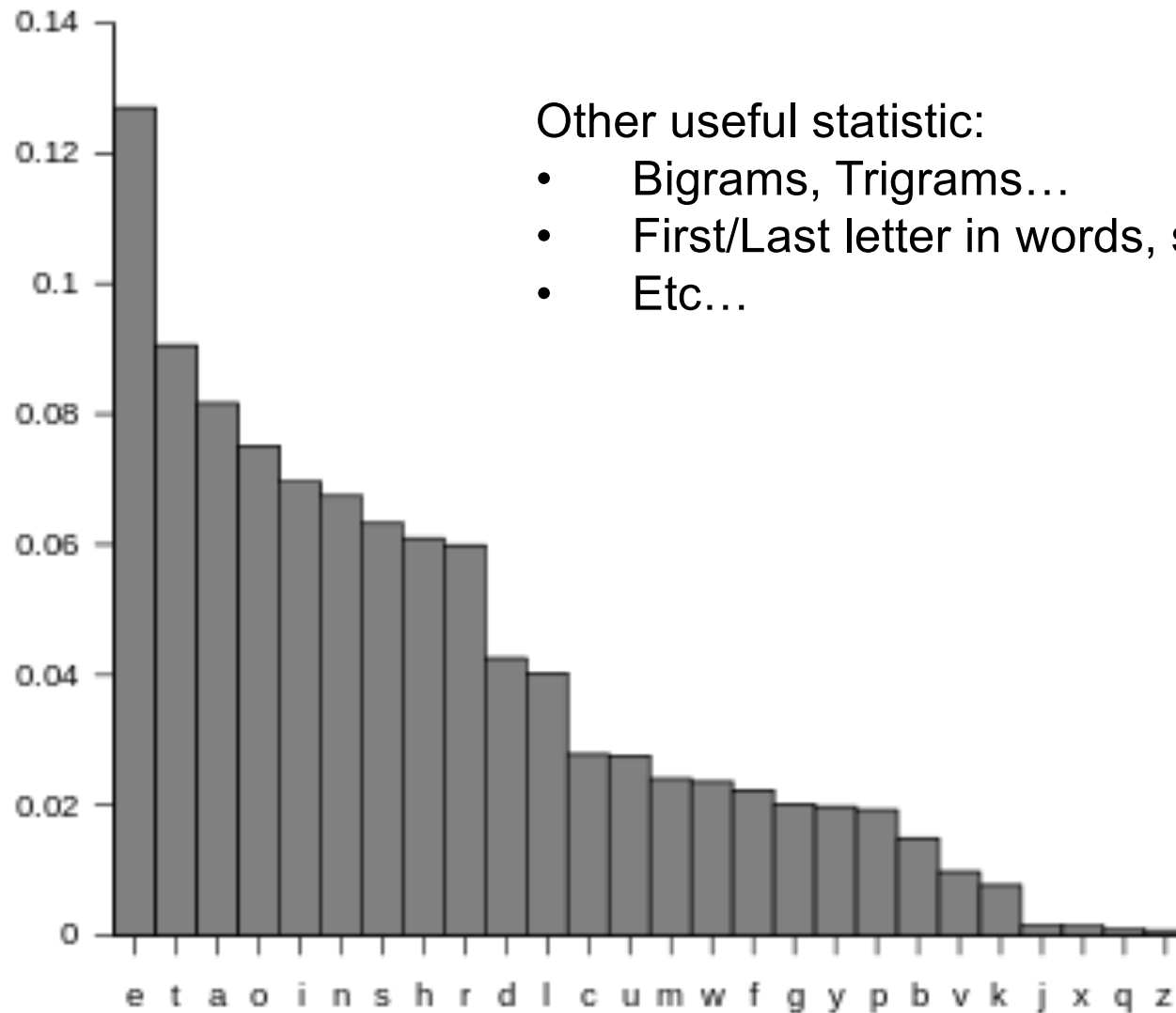
# Monoalphabetic Substitution Ciphers

- Generalize Caesar and AzBy:
  - Other permutations of letters
    - To letters or to other symbols (no real difference)
  - Keyed: Given key  $k$ , cipher  $E_k$  is a permutation
  - Or: the 'key' is simply the permutation (table)
  - Classical, 'elementary school' cryptosystem
- Examples:



- **Vulnerable to letter-frequency cryptanalysis**

# Letter frequencies (in English)



Other useful statistic:

- Bigrams, Trigrams...
- First/Last letter in words, sentences
- Etc...

# Example: Frequency Cryptanalysis

## Given ciphertext:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAI Z  
VUEPHZHMDZSHZOWSFPAPPDTSVPPQUZWYMXUZUHSX  
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

## Count relative letter frequencies:

A	B	C	D	E	F	G	H	I	J	K	L	M
2	2	0	6	6	4	2	7	1	1	0	0	8
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	9	16	3	0	10	3	10	5	4	5	2	14

# Example: Frequency Cryptanalysis

## Given ciphertext:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ  
VUEPHZHMDZSHZOWSFPAPPDTSVPPQUZWYMXUZUHSX  
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t											
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

Most frequent letter is e, so: P=E(e)

Second frequent is t, so: Z=E(t) ... let's replace...

# Example: Frequency Cryptanalysis

## Given ciphertext:

U t Q S O V U O H X M O e V G e O t e E V S G t W S t O e F e E S X U D B M E T S X A I t  
V U E e H t H M D t S H t O W S F e A e e D T S V e Q U t W Y M X U t U H S X  
E e Y E e O e D t S t U F e O M B t W e F U e t H M D J U D T M O H M Q

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t											
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

In English texts, `t` is often followed by `h`. Count chars following Z (t):  
Twice: W, H, U and O; once: Q, V, D & S. **Pick W, since this gives `the`...**

# Example: Frequency Cryptanalysis

## Given ciphertext:

UtQSOVUOHXMOeVGteEVSGthStOeFeESXUDBMETSXAIIt  
VUEeHtHMDtSHtohSFeAeeDTSVeQUthYMXUtUHSX  
EeYEeOedtStUFeOMBtheFUetHMDJUDTMOHMQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a										h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

We have **thSt** with S being third-most common. After **e** and **t**, most common letters are: **aoins**hr (in this order). Only **`a`** fits, so...

# Example: Frequency Cryptanalysis

## Given ciphertext:

UtQaOVUOHXMOeVGeoteEVaGthatoeFeEaXUDBMETaXAIIt  
VUEeHtHMDtaHtOhaFeAeedTaVeQUthYMXUtUHax  
EeYEeOedtatuFOMBtheFUethHMDJUDTMOHMQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a										h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

Next common in ciphertext is U and in English are **oinshr** (in this order).

Few, rare words begin with `ot` (and not `oth`), but `it` is common, so: U=E(i) !

# Example: Frequency Cryptanalysis

## Given ciphertext:

itQaOViOHXMOeVGeOteEVaGthatOeFeEaXiDBMETaXAIt  
ViEeHtHMDtaHtOhaFeAeeDTaVeQithYMXitiHaX  
EeYEeOeDtatiFeOMBtheFietHMDJiDTMOHMQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i									h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

Next common in ciphertext are OMH and in English are **onsr** (in this order).  
'O'=E('o') is unlikely since it gives `that **oeFeEa...**` → try 'M'=E('o')...



# Example: Frequency Cryptanalysis

## Given ciphertext:

itQaOViOHXoOeVGeOteEvaGthatOeFeEaXiDBoETaXAIt  
ViEeHtHoDtahTohaFeAeeDTaVeQithYoXitiHaX  
EeYEeOeDtatiFeOoBtheFiethoDJiDToOHoQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i		o							h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

Next common in ciphertext is O and in English is s... go for it: O=E(s)!

# Example: Frequency Cryptanalysis

## Given ciphertext:

itQasVisHXoseVGesteEvaG that seFeEaXiDBoETaXAI t  
ViEeHtHoDtahTshaFeAeeDTaVeQithYoXitiHaX  
EeYEeseDtatiFesoBtheFiethoDJiDTosHoQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o							h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0

`that' is mostly one word. Most common last-letter not assigned yet is `y`, which is not a common word, so:  $G = E(y)$ ...

# Example: Frequency Cryptanalysis

Given ciphertext:

itQasVisHXoseV~~vesteEVay~~ that seFeEaXiDBoETaXAIt  
ViEeHtHoDtahTshaFeAeeDTaVeQithYoXitiHaX  
EeYEeseDtatiFesoBtheFiethoDJiDTosHoQ

Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o							h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
				y								

We now simply recognize the (quite common) word `yesterday', so:  
 $E=E(r)$ ,  $V=E(d)$ ...

# Example: Frequency Cryptanalysis

## Given ciphertext:

itQasdisHXosed yesterday that seFeraXiDBorTaXAIt  
direHtHoDtataHtshaFeAeeDTadeQithYoXitiHaX  
reYreseDtatiFesoBtheFiethoDJiDTosHoQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o			r	d			h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
				y								

Next unused common letter is **n** (by far). But H doesn't seem to fit... so D=E(**n**)...

# Example: Frequency Cryptanalysis

Given ciphertext:

itQasdisHXosed yesterday that seFeraXinBorTaXAIt  
direHtHontaHtshaFeAeenTadeQithYoXitiHaX  
reYresentatiFesoBtheFietHonJinTosHoQ

Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o		n	r	d			h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
				y								

Long string with only one cipher-letter, H... only c fits so: H=E(c)...

# Example: Frequency Cryptanalysis

## Given ciphertext:

itQasdiscXosed yesterday that seFeraXinBorTaXAIt  
direct contacts haFeAeenTadeQithYoXiticaX  
reYresentatiFesoBtheFietconJinToscoQ

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o	c	n	r	d			h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
				y								

Next common cipher-letter is X and plain-letter is **l**, and it indeed fits: X=E(**l**) !

# Example: Frequency Cryptanalysis

Given ciphertext:

itQas disclosed yesterday that seFeralinBorTalAIt  
direct contacts haFeAeenTadeQithYolitical  
reYresentatiFesoBtheFietconJinToscoQ

Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o	c	n	r	d	l		h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
				y								

Next identify text begins with `it was' and also two quite common words  
so : Q=E(w), Y=E(p), F=E(v) !

# Example: Frequency Cryptanalysis

Given ciphertext:

it was disclosed yesterday that several inBorTalAIt  
direct contacts have AeenTade with political  
representatives oBthevietconJinToscow

Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o	c	n	r	d	l	v	h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
w				y	p							

Next: `oB'->`of', `Aeen'->been, `Tade'->made, `vietconJ'->Vietcong, ..



# Example: Frequency Cryptanalysis

## Given ciphertext:

it was disclosed yesterday that several informal bIt  
direct contacts have been made with political  
representatives of the vietcong in moscow

## Sorted:

P	Z	S	U	O	M	H	D	E	V	X	F	W
16	14	10	10	9	8	7	6	6	5	5	4	4
e	t	a	i	s	o	c	n	r	d	l	v	h
Q	T	A	B	G	Y	I	J	C	K	L	N	R
3	3	2	2	2	2	1	1	0	0	0	0	0
w	m	b	f	y	p		g					

(finally:  $I=E(u)$  )

---

# Security-by-Obscurity Ciphers

- Previous ciphers' security relied on obscurity
  - I.e., hope attacker does not know cipher
- Used extensively until 1883
  - ... but usually cryptanalyzed
  - ... esp. after encryption devices were captured
- What happened in 1883??
  - A conceptual leap in cryptography and security

---

# Kerckhoffs' Known Design Principle [1883]

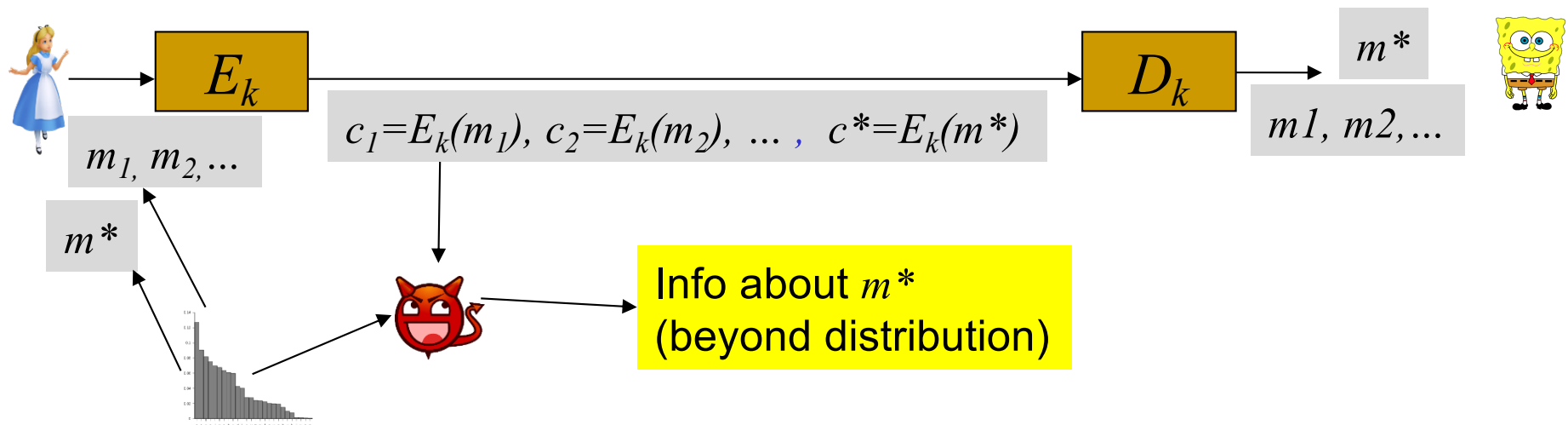
- Assume adversary knows the design – everything except the secret keys
- No `security by obscurity'
  - Although attacking obscure design is harder
- Why assume/use public design ?
  - No need to replace system once design is exposed
  - Usually stronger
  - Establish standards for multiple applications:
    - Efficiency of production and of test attacks / cryptanalysis
- Secrecy is based only on secrecy of key

# Exhaustive Key Search

- Kerckhoffs:  $\text{Secrecy} \leq \text{secrecy of key } k$
- **Exhaustive Key Search**: try all keys  $k' \in \{0,1\}^{|k|}$
- How to identify correct key  $k = k' ??$
- Depends on **attacker capability (model)**
  - Critical element of security analysis!!
  - Attack models we will study:
    - Cipher-Text Only (CTO) attack
    - Known-plaintext attack (KPA)
    - Chosen-plaintext attack (CPA)
    - Chosen-ciphertext attack (CCA)

# Cipher-Text Only (CTO) attack

- Adversary have previous knowledge about all possible plaintexts, like their distribution.
- Attacker can infer info about the challenge plaintext  $m^*$  beyond the initial info it has.
  - This is given only ciphertexts and the plaintext distribution

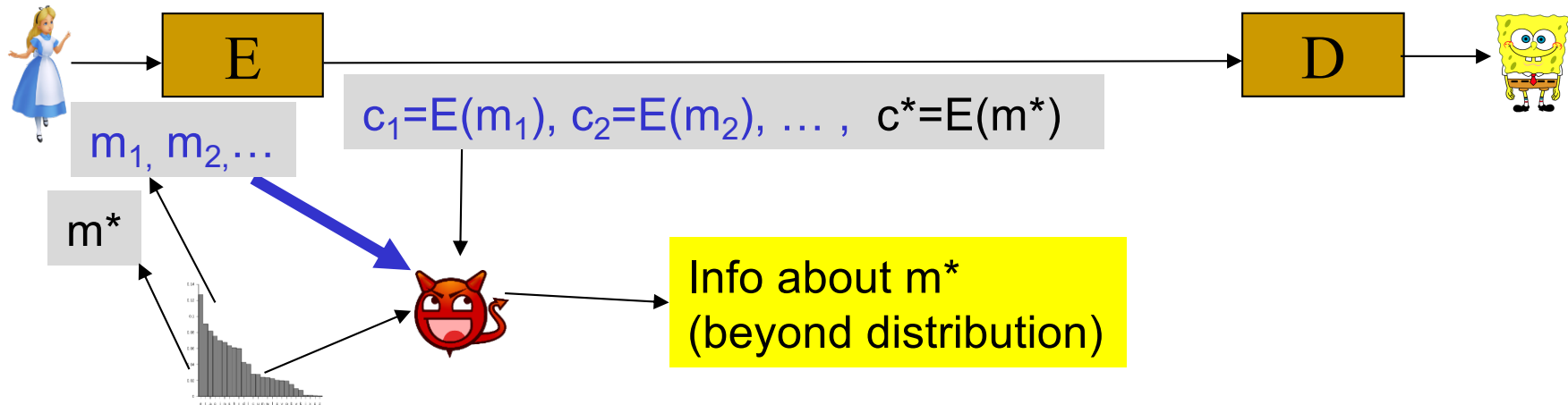


# Exhaustive Key Search and CTO

- **Exhaustive Key Search:** try all keys  $k' \in \{0,1\}^{|k|}$
- How to identify correct key  $k = k'$  given CTO??
  - Decrypt ciphertexts, then check resulting 'plaintext'
    - Let  $m_1, m_2, \dots$  be a set of random plaintext samples
    - Let  $c_1 = E_k(m_1), c_2 = E_k(m_1), \dots$  be corresponding ciphertexts
    - To test if the key is  $k'$ , compute set  $M' = \{D_{k'}(c_1), D_{k'}(c_2), \dots\}$
    - If  $M'$  fits plaintext distribution:  $k'$  is probably the key
    - Otherwise:  $k'$  is probably not the key
  - Challenge: test often is inconclusive

# Known Plaintext Attack (KPA)

- Sample messages  $M=\{m_1, m_2, \dots\}$  from a given distribution.
- Give M and ciphertexts  $c_1=E(m_1), c_2, \dots$  to attacker



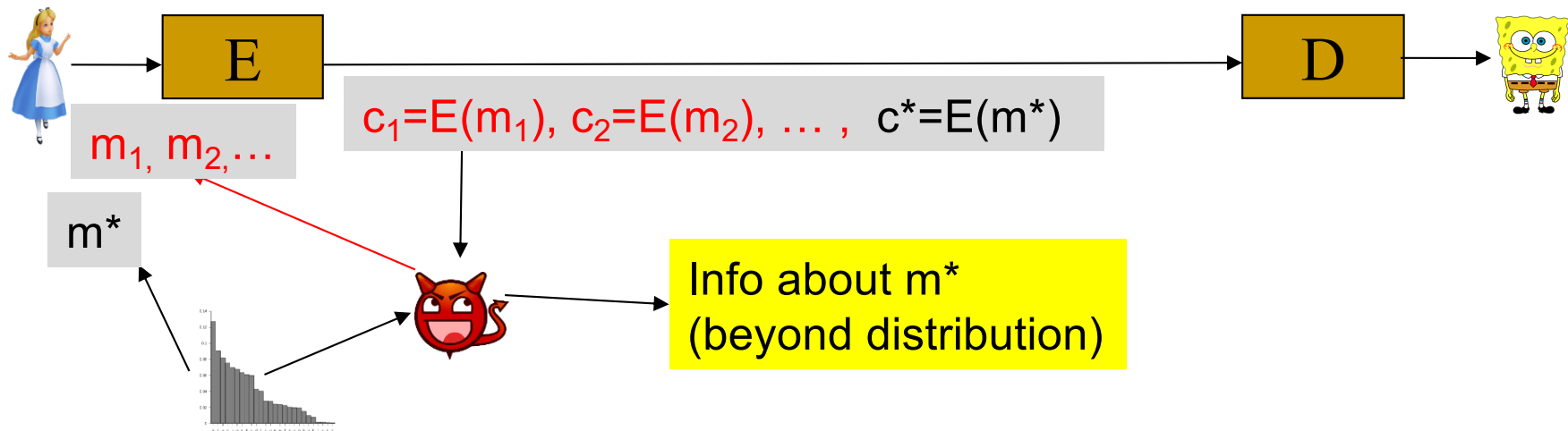
# Exhaustive Key Search and KPA

- **Exhaustive Key Search:** try all keys  $k' \in \{0,1\}^{|k|}$
- How to identify correct key  $k = k'$  given KPA??
  - Attacker obtains known plaintext, ciphertext pairs:  $(m_1, c_1=E_k(m_1)), (m_2, c_2=E_k(m_2)), \dots$
  - To test if the key is  $k'$ , compute  $m'_1=D_{k'}(c_1), m'_2=D_{k'}(c_2), \dots$
  - If for every pair  $i$  holds  $m'_i=m_i$  then  $k'$  is probably the key
  - Otherwise:  $k'$  is probably not the key
- CTO and KPA attacks must test about half the keys.
  - On average, the attacker will find the key after trying half of all possible keys.



# Chosen Plaintext Attack (CPA)

- Beside the plaintext distribution/initial info, attacker can choose messages  $m_1, m_2, \dots$
- Give ciphertexts of these plaintext messages to attacker.



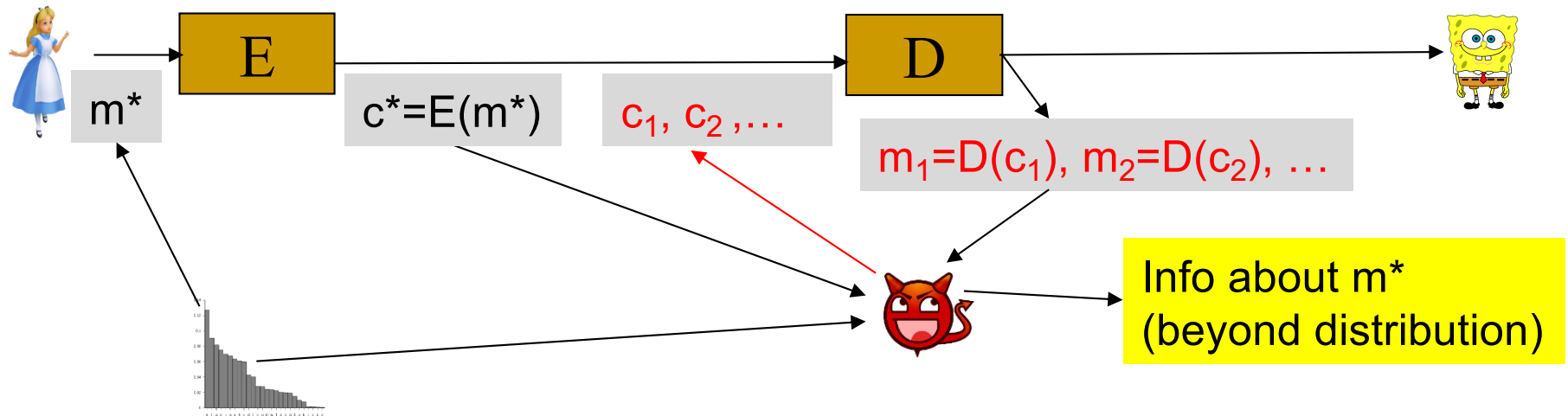
# Exhaustive Key Search and CPA

## ■ Generic CPA: Table-Lookup

- Choose some fixed plaintext  $m$ 
  - E.g., some default message: `good morning!`
  - Quite common in practice... e.g., in web (http), GSM,...
- Offline: fill a table  $T$ . For every key  $k'$ , compute  $T(k') = E_{k'}(m)$
- Online: select plaintext  $m$ , obtain  $c = E_k(m)$
- If  $T(k') = c$  then  $k'$  probably the key:  $k' = k$
- Otherwise:  $k'$  is probably not the key
- Time complexity  $t = O(1)$  lookup time, requires  $2^{|k|}$  memory
- More advanced: Time/Memory tradeoffs (e.g., rainbow tables)
  - Use hash-functions, so we can't yet discuss

# Chosen Ciphertext Attack (CCA)

- Attacker can select ciphertexts  $c_1, c_2, \dots$
- And receive decryptions:



---

# The Attack Models Championship

- We discussed several attack models:
    - CTO, KPA, CPA, CCA
  - Model A is stronger than model B, if a cipher secure against A is also secure against B
    - Notation:  $A > B$
    - Example:  $KPA > CTO$  [why?]
  - KPA vs. CPA ?
  - KPA vs. CCA?
  - CPA vs. CCA ?
-

# Sufficient Effective Key Length

## ■ *Sufficient Effective Key Length Principle:*

- ❑ Keys should be long enough to make attacks infeasible, for best adversary resources expected, during `sensitivity period` of data
- ❑ Exhaustive search – or other attacks
- ❑ *Large key-space is necessary, but not sufficient*
  - ❑ Monoalphabetic substitution cipher, with permutation as key:  $26! = 4 \cdot 10^{26}$  keys... yet insecure!
  - ❑ *Effective key length:* log of number of trials by the most effective attack
    - ❑ Same as **number of bits for exhaustive search**
    - ❑ Defined for specific attack models

---

# Thank You!

