Cryptography - *Day 3*

Defining Security

Review

XOR Operation

- XOR is a binary "exclusive or" operation that is represented by $\oplus$
- Suppose $A = a_1 \ldots a_n$ and $B = b_1 \ldots b_n$ then A $\oplus$ B = C where C= $c_1 \ldots c_n$ such that

  c_i = 0 if $a_i = b_i$ and

  $c_i = 1$ if $a_i \neq b_i$.

XOR Operation

- Suppose $A = 1001\ 0010$ and $B = 0000\ 1110$

- A $\oplus$ B = $\quad 1001\ 0010$

$$\oplus\ 0000\ 1110$$

$$1001\ 1100$$

Byte-wise shift cipher

- $\mathcal{M}$ = {strings of bytes}
- Gen: choose uniform byte $k \in \mathcal{K} = \{0, \ldots, 255\}$
- $\text{Enc}_k(m_1 \ldots m_t)$: output $c_1 \ldots c_t$, where

  $$c_i := m_i \oplus k$$

- $\text{Dec}_k(c_1 \ldots c_t)$: output $m_1 \ldots m_t$, where

  $$m_i := c_i \oplus k$$

Example

- Say plaintext is “Hi” and key is 1111 0001
- “Hi” = 0x48 69 = 0100 1000 0110 1001
- XOR with “Hi” with the key
- 0100 1000 0110 1001 $\oplus$
  1111 0001 1111 0001
  = 1011 1001 1001 1000=0xB9 98=unprintable

Byte-wise Vigenère cipher

- The key is a string of bytes
- The plaintext is a string of bytes
- To encrypt, XOR each character in the plaintext with the next character of the key
  - Wrap around in the key as needed
- Decryption just reverses the process

Example

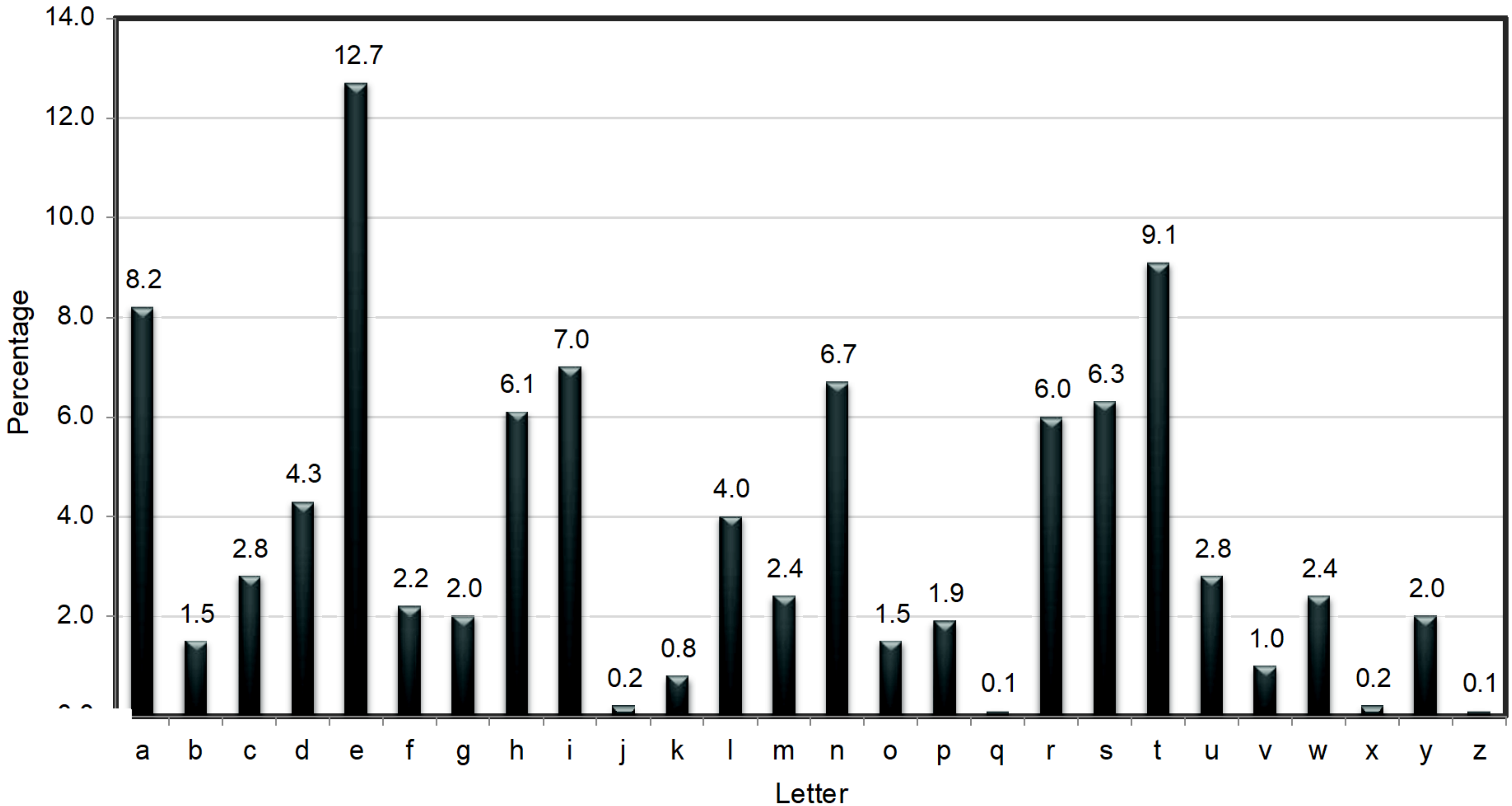
- Say plaintext is “Hello!” and key is 0xA1 2F
- “Hello!” = 0x48 65 6C 6C 6F 21
- XOR with 0xA1 2F A1 2F A1 2F
- 0x48 $\oplus$ 0xA1
  - 0100 1000 $\oplus$ 1010 0001 = 1110 1001 = 0xE9

- Ciphertext: 0xE9 4A CD 43 CE 0E

Attacking the (variant) Vigenère cipher

- Two steps:
  - Determine the key length
  - Determine each byte of the key

Using plaintext letter frequencies



Useful observations

- Only 128 valid ASCII chars (128 bytes invalid)
- 0x20-0x7E printable
- 0x41-0x7a includes upper/lowercase letters
  - Uppercase letters begin with 0x4 or 0x5
  - Lowercase letters begin with 0x6 or 0x7

Determining the key length

- Let p_i (for $0 \leq i \leq 255$) be the frequency of **byte** i in general English text
  - I.e., $p_i = 0$ for $i < 32$ or $i > 127$
  - I.e., p_{97} = frequency of ‘a’
  - The distribution is far from uniform

Determining the key length

- If the key length is N, then every N^{th} character of the plaintext is encrypted using the same "shift"
  - If we take every N^{th} character and calculate frequencies, we should get the p_i's in permuted order
  - If we take every M^{th} character (M not a multiple of N) and calculate frequencies, we should get something close to uniform

Determining the key length

- Assume length is k
- For key length k, tabulate $q_0, \ldots, q_{255}$ and compute $\Sigma\, q_i^2$
  - If close to uniform, $\Sigma\, q_i^2 \approx 256 \cdot (1/256)^2 = 1/256$
  - If a permutation of p_i, then $\Sigma\, q_i^2 \approx \Sigma\, p_i^2$
    - Could compute $\Sigma\, p_i^2$ (but somewhat difficult)
    - Key point: will be much larger than 1/256
- Compute $\Sigma\, q_i^2$ for each possible key length, and look for maximum value

Determining the i^{th} byte of the key

- Assume the key length N is known
- Look at every N^{th} character of the ciphertext, starting with the i^{th} character
  - Call this the i^{th} ciphertext "stream"
  - Note that all bytes in this stream were generated by XORing plaintext with the same byte of the key
- Try decrypting the stream using every possible byte value B
  - Get a candidate plaintext stream for each value

Determining the i^{th} byte of the key

- Could use $\{p_i\}$ as before, but not easy to find
- When the guess B is correct:
  - Frequencies of lowercase letters (as a fraction of all lowercase letters) should be close to known English-letter frequencies
    - Tabulate observed letter frequencies $q'_0, \ldots, q'_{25}$ (as fraction of all lowercase letters)
    - Should find $\Sigma\, q'_i\, p'_i \approx \Sigma\, p'^2_i \approx 0.065$, where p'_i corresponds to English-letter frequencies
    - In practice, take B that maximizes $\Sigma\, q'_i\, p'_i$

Defining secure encryption

Crypto definitions (generally)

- Security guarantee/goal
  - What we want to achieve and/or what we want to prevent the attacker from achieving
- Threat model
  - What (real-world) capabilities the attacker is assumed to have

Recall

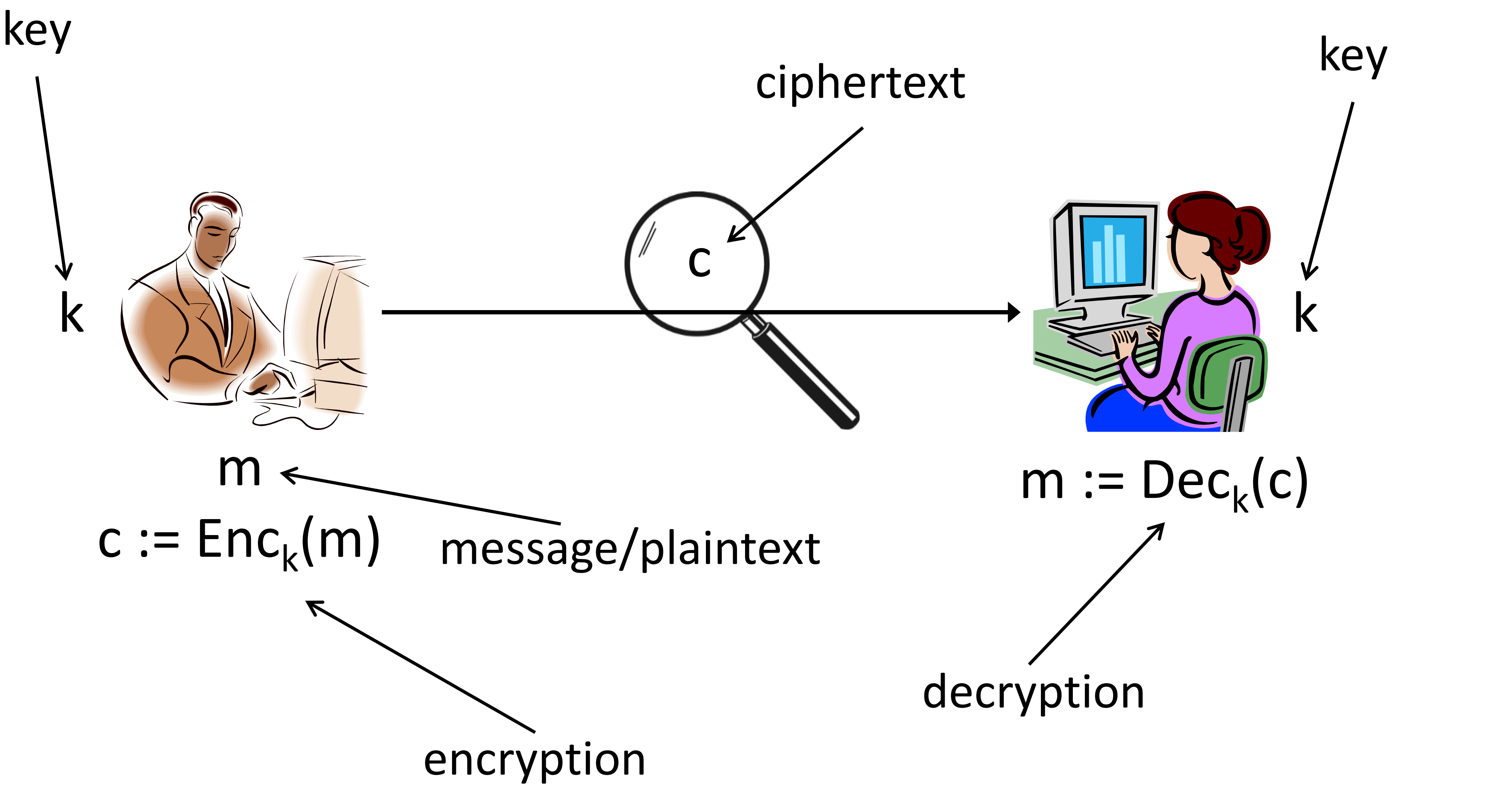
- A *private-key encryption scheme* is defined by a message space $\mathcal{M}$ and algorithms (Gen, Enc, Dec):
  - Gen (key-generation algorithm): generates k
  - Enc (encryption algorithm): takes key k and message $m \in \mathcal{M}$ as input; outputs ciphertext c

    $$c \leftarrow \text{Enc}_k(m)$$

  - Dec (decryption algorithm): takes key k and ciphertext c as input; outputs m.

    $$m := \text{Dec}_k(c)$$

Private-key encryption



Threat models for encryption

- Ciphertext-only attack - obtain only ciphertext
- Known-plaintext attack – obtain ciphertext with some knowledge of the message
- Chosen-plaintext attack - obtain encryptions of chosen messages
- Chosen-ciphertext attack – obtain decryptions of chosen ciphertext

Goal of secure encryption?

- How would you define what it means for encryption scheme (Gen, Enc, Dec) over message space $\mathcal{M}$ to be secure?
  - Against a (single) ciphertext-only attack

Secure encryption?

- “Impossible for the attacker to learn the plaintext from the ciphertext”
  - What if the attacker learns 90% of the plaintext?

Secure encryption?

- "Impossible for the attacker to learn any character of the plaintext from the ciphertext"
  - What if the attacker is able to learn (other) partial information about the plaintext?
    - E.g., salary is greater than $75K