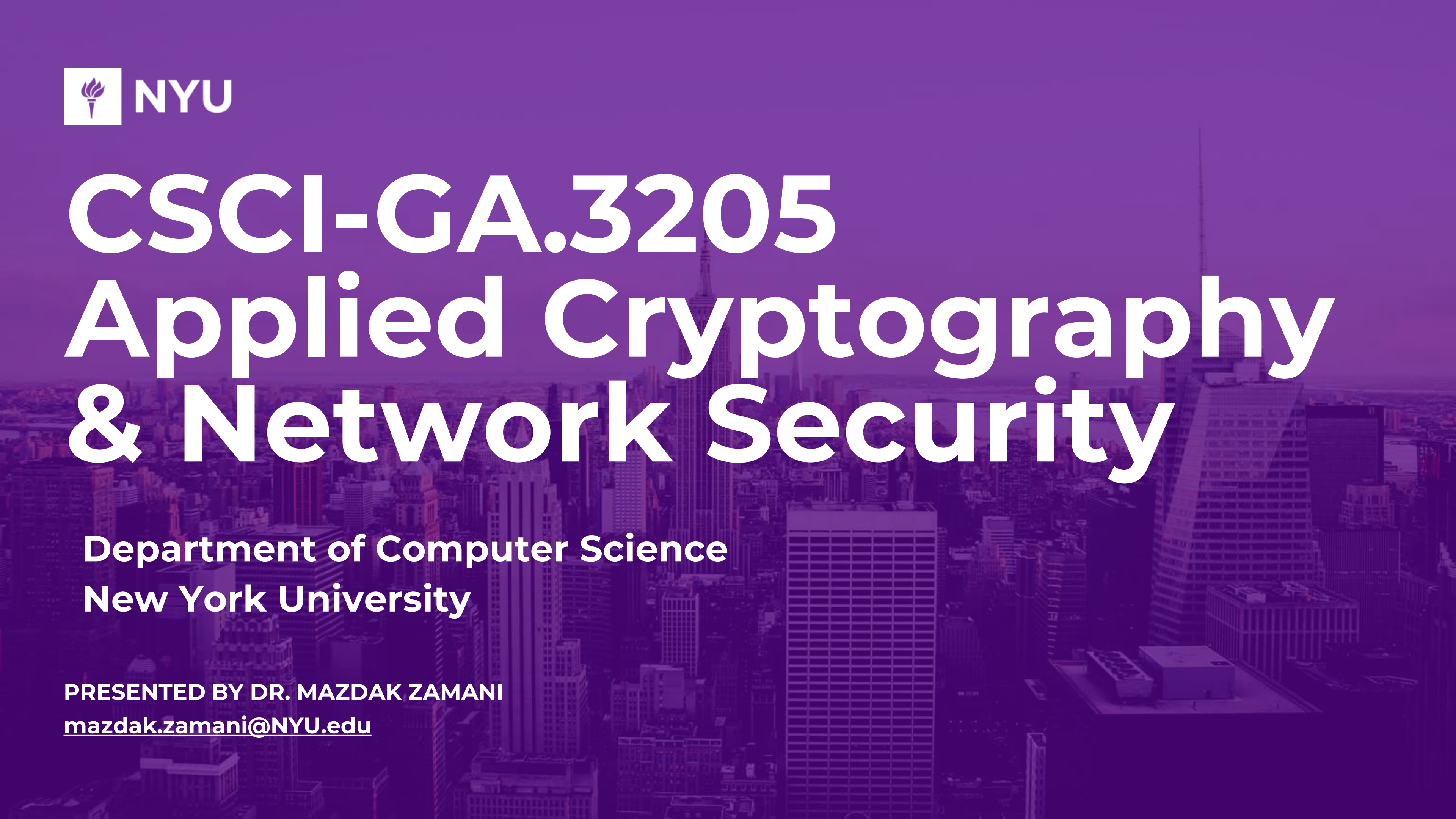


NYU

CSCI-GA.3205 Applied Cryptography & Network Security

Department of Computer Science
New York University

PRESENTED BY DR. MAZDAK ZAMANI
mazdak.zamani@NYU.edu

Chapter 1
Introduction

01

NYU

Modern Cryptography

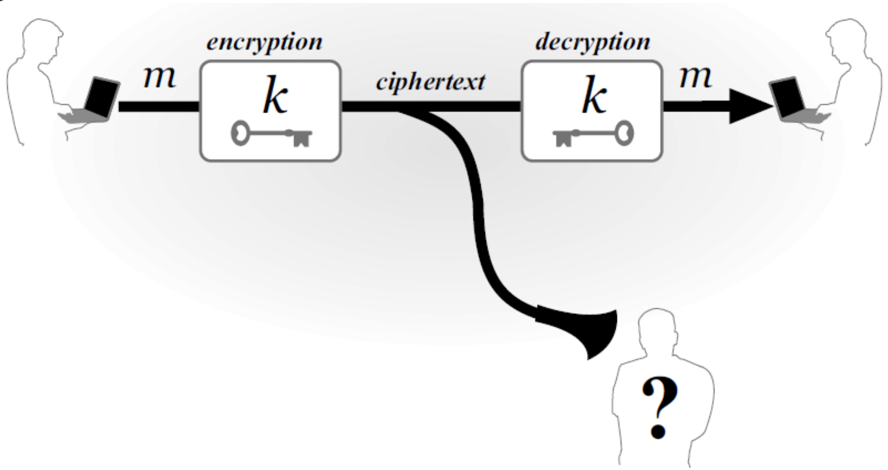
- The Concise Oxford English Dictionary defines cryptography as “the art of writing or solving codes.”
    - The definition focuses solely on the codes that have been used for centuries to enable secret communication.
    - But cryptography nowadays encompasses much more than this: it deals with mechanisms for ensuring integrity, techniques for exchanging secret keys, protocols for authenticating users, electronic auctions and elections, digital cash, and more.
- Modern cryptography involves the study of mathematical techniques for securing digital information, systems, and distributed computations against adversarial attacks.

The Setting of Private-Key Encryption

- Classical cryptography was concerned with designing and using *codes* (also called *ciphers*) that enable two parties to communicate secretly in the presence of an eavesdropper who can monitor all communication between them.
- In modern parlance, codes are called encryption schemes and that is the terminology we will use here.
- Security of all classical encryption schemes relied on a secret—*a key*—shared by the communicating parties in advance and unknown to the eavesdropper. This scenario is known as the *private-key* (or *shared-/secret-key*) setting.

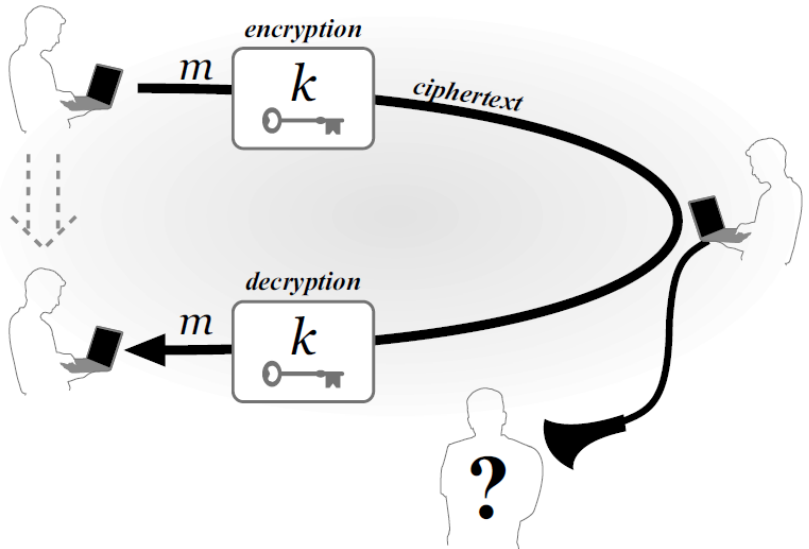
The Setting of Private-Key Encryption – Cont'd

- Two parties share a key and use this key when they want to communicate secretly.
- One party can send a message, or *plaintext*, to the other by using the shared key to *encrypt* (or "scramble") the message and thus obtain a *ciphertext* that is transmitted to the receiver.
- The receiver uses the same key to *decrypt* (or "unscramble") the ciphertext and recover the original message.



Two canonical applications of private-key cryptography

- There are two canonical applications of private-key cryptography.
  - In the first, there are two distinct parties separated in *space*.
  - The second widespread application of private-key cryptography involves the same party communicating with itself over *time*.

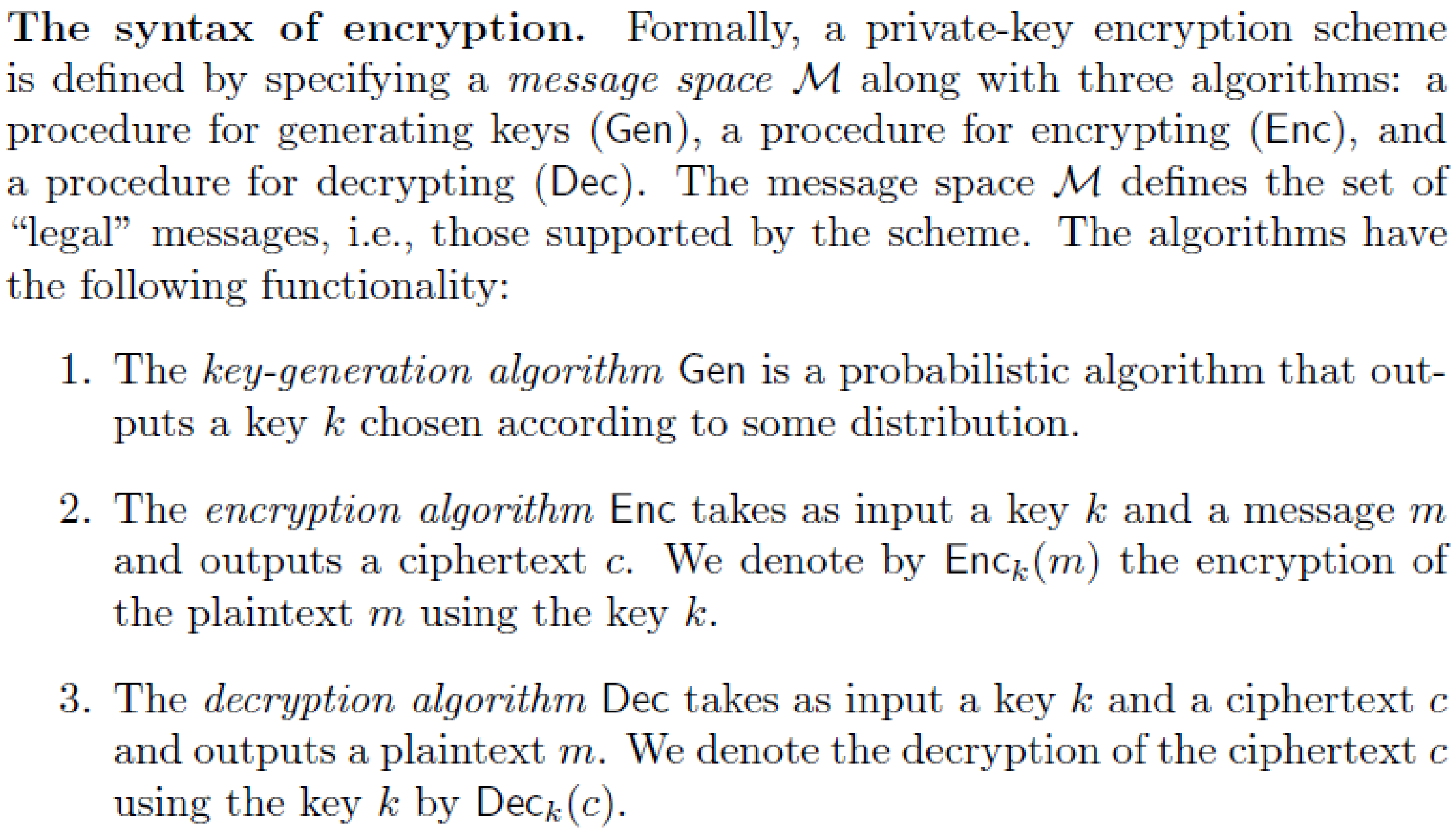


Symmetric vs Asymmetric

- In the *symmetric-key* setting the same key is used to convert the plaintext into a ciphertext and back.
- The symmetry lies in the fact that both parties hold the same key that is used for encryption and decryption.
- This is in contrast to *asymmetric*, or *public-key*, encryption, where encryption and decryption use different keys.

NYU

The syntax of encryption



The syntax of encryption. Formally, a private-key encryption scheme is defined by specifying a *message space* $\mathcal{M}$ along with three algorithms: a procedure for generating keys (Gen), a procedure for encrypting (Enc), and a procedure for decrypting (Dec). The message space $\mathcal{M}$ defines the set of "legal" messages, i.e., those supported by the scheme. The algorithms have the following functionality:

1. The *key-generation algorithm* Gen is a probabilistic algorithm that outputs a key k chosen according to some distribution.

2. The *encryption algorithm* Enc takes as input a key k and a message m and outputs a ciphertext c. We denote by $\mathsf{Enc}_k(m)$ the encryption of the plaintext m using the key k.

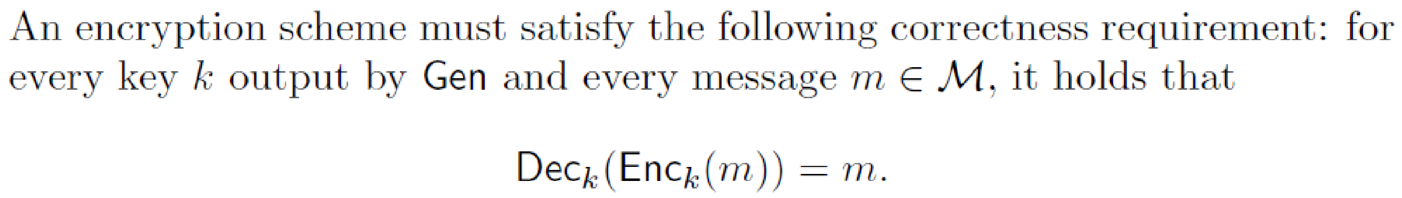
3. The *decryption algorithm* Dec takes as input a key k and a ciphertext c and outputs a plaintext m. We denote the decryption of the ciphertext c using the key k by $\mathsf{Dec}_k(c)$.

Correctness requirement

Encrypting a message and then decrypting the resulting ciphertext (using the same key) yields the original message.

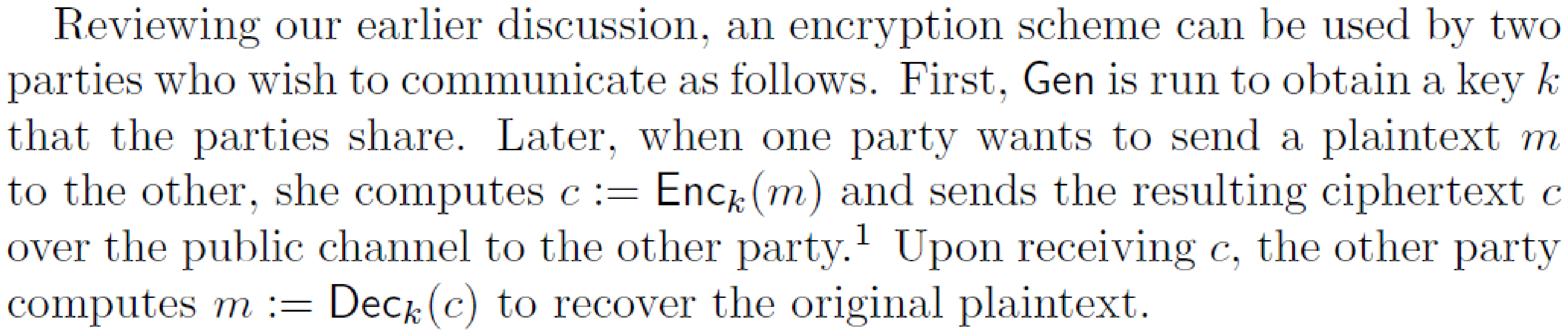
The set of all possible keys output by the key-generation algorithm is called the key space and is denoted by **K**.

Almost always, **Gen** simply chooses a uniform key from the key space;


An encryption scheme must satisfy the following correctness requirement: for every key k output by Gen and every message $m \in \mathcal{M}$, it holds that

$$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m.$$

Encryption scheme



Reviewing our earlier discussion, an encryption scheme can be used by two parties who wish to communicate as follows. First, Gen is run to obtain a key k that the parties share. Later, when one party wants to send a plaintext m to the other, she computes $c := \mathsf{Enc}_k(m)$ and sends the resulting ciphertext c over the public channel to the other party.[1] Upon receiving c, the other party computes $m := \mathsf{Dec}_k(c)$ to recover the original plaintext.

[1] We use ":=" to denote deterministic assignment and assume for now that Enc is deterministic.

Keys and Kerckhoffs’ principle

- Kerckhoffs’ principle:
  - The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.
- That is, an encryption scheme should be designed to be secure even if an eavesdropper knows all the details of the scheme, so long as the attacker doesn’t know the key being used.
- Stated differently, security should not rely on the encryption scheme being secret; instead, Kerckhoffs’ principle demands that security rely solely on secrecy of the key.
- There are three primary arguments in favour of Kerckhoffs’ principle.
  - It is significantly easier for the parties to maintain secrecy of a short key than to keep secret the (more complicated) algorithm they are using.
  - In case the honest parties’ shared, secret information is ever exposed, it will be much easier for them to change a key than to replace an encryption scheme.
  - For large-scale deployment it is significantly easier for users to all rely on the same encryption algorithm/software (with different keys) than for everyone to use their own custom algorithm.

Historical Ciphers

- To highlight the weaknesses of an “ad hoc” approach to cryptography
- To demonstrate that simple approaches to achieving secure encryption are unlikely to succeed

- Plaintext characters are written in lower case and ciphertext characters are written in UPPER CASE for typographical clarity.

Caesar's cipher

- Julius Caesar encrypted by shifting the letters of the alphabet 3 places forward: a was replaced with D, b with E, and so on.
- At the very end of the alphabet, the letters wrap around and so z was replaced with C, y with B, and x with A.
- For example, encryption of the message **begin the attack now**, with spaces removed, gives: **EHJLQWKHDWWDFNQRZ**.

- An immediate problem with this cipher is that the encryption method is fixed; there is no key.
- Thus, anyone learning how Caesar encrypted his messages would be able to decrypt effortlessly.

The shift cipher

- The *shift cipher* can be viewed as a keyed variant of Caesar's cipher.
- In the shift cipher the key k is a number between 0 and 25.
- To encrypt, letters are shifted as in Caesar's cipher, but now by ***k*** places.
- Mapping this to the syntax of encryption described earlier, the message space consists of arbitrary length strings of English letters with punctuation, spaces, and numerals removed, and with no distinction between upper and lower case.
- Algorithm **Gen** outputs a uniform key **k** $\in \{0, \ldots, 25\}$
- Algorithm **Enc** takes a key **k** and a plaintext and shifts each letter of the plaintext *forward* **k** positions
- Algorithm **Dec** takes a key **k** and a ciphertext and shifts every letter of the ciphertext *backward* **k** positions

Mathematical description of shift cipher

A more mathematical description is obtained by equating the English alphabet with the set $\{0, \ldots, 25\}$ (so $\mathtt{a} = 0$, $\mathtt{b} = 1$, etc.). The message space $\mathcal{M}$ is then any finite sequence of integers from this set. Encryption of the message $m = m_1 \cdots m_\ell$ (where $m_i \in \{0, \ldots, 25\}$) using key k is given by

$$\mathsf{Enc}_k(m_1 \cdots m_\ell) = c_1 \cdots c_\ell, \quad \text{where } c_i = [(m_i + k) \bmod 26].$$

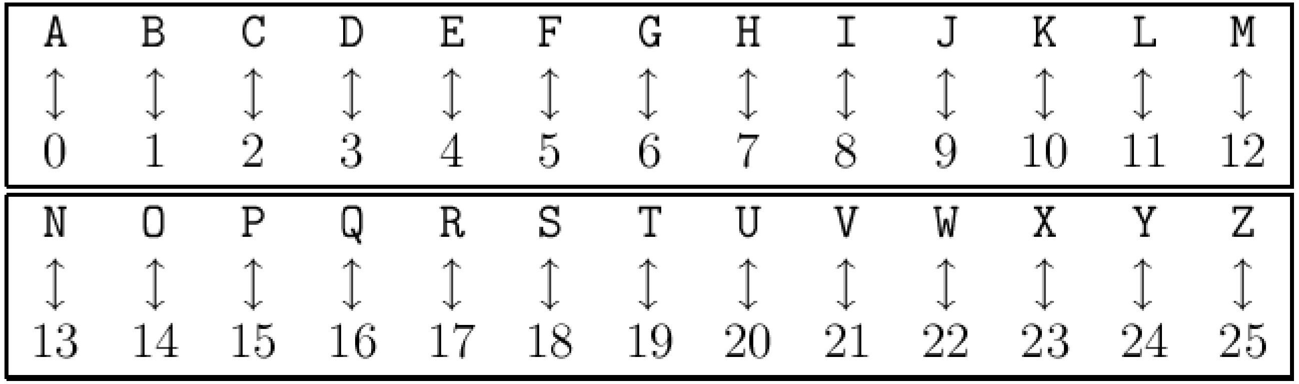
(The notation $[a \bmod N]$ denotes the remainder of a upon division by N, with $0 \leq [a \bmod N] < N$. We refer to the process mapping a to $[a \bmod N]$ as *reduction modulo* N; we will have more to say about this beginning in Chapter 8.) Decryption of a ciphertext $c = c_1 \cdots c_\ell$ using key k is given by

$$\mathsf{Dec}_k(c_1 \cdots c_\ell) = m_1 \cdots m_\ell, \quad \text{where } m_i = [(c_i - k) \bmod 26].$$

Is the shift cipher secure?

- Try to decrypt the following ciphertext that was generated using the shift cipher and a secret key k:

  OVDTHUFWVZZPISLRLFZHYLAOLYL.
- Is it possible to recover the message without knowing k?



A	B	C	D	E	F	G	H	I	J	K	L	M
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
13	14	15	16	17	18	19	20	21	22	23	24	25

Sufficient key-space principle

- An attack that involves trying every possible key is called a *brute-force* or *exhaustive-search* attack. Clearly, for an encryption scheme to be secure it must not be vulnerable to such an attack. This observation is known as the *sufficient key-space principle*:
- *Any secure encryption scheme must have a key space that is sufficiently large to make an exhaustive-search attack infeasible.*
- One can debate what amount of effort makes a task "infeasible," and an exact determination of feasibility depends on both the **resources** of a potential attacker and the length of **time** the sender and receiver want to ensure secrecy of their communication.
- Nowadays, attackers can use supercomputers, tens of thousands of personal computers, or graphics processing units (GPUs) to speed up brute-force attacks.
- To protect against such attacks the key space must therefore be very large—say, of size at least 2^{70}, and even larger if one is concerned about long-term security against a well-funded attacker.
- The sufficient key-space principle gives a *necessary* condition for security, but not a *sufficient* one.

Mono-alphabetic substitution cipher

- In the mono-alphabetic substitution cipher, the key also defines a map on the alphabet, but the map is now allowed to be arbitrary subject only to the constraint that it be one-to-one so that decryption is possible.
- The keys pace thus consists of all bijections, or permutations, of the alphabet.
- For example, the key that defines the following permutation

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
X	E	U	A	D	N	B	K	V	M	R	O	C	Q	F	S	Y	H	W	G	L	Z	I	J	P	T

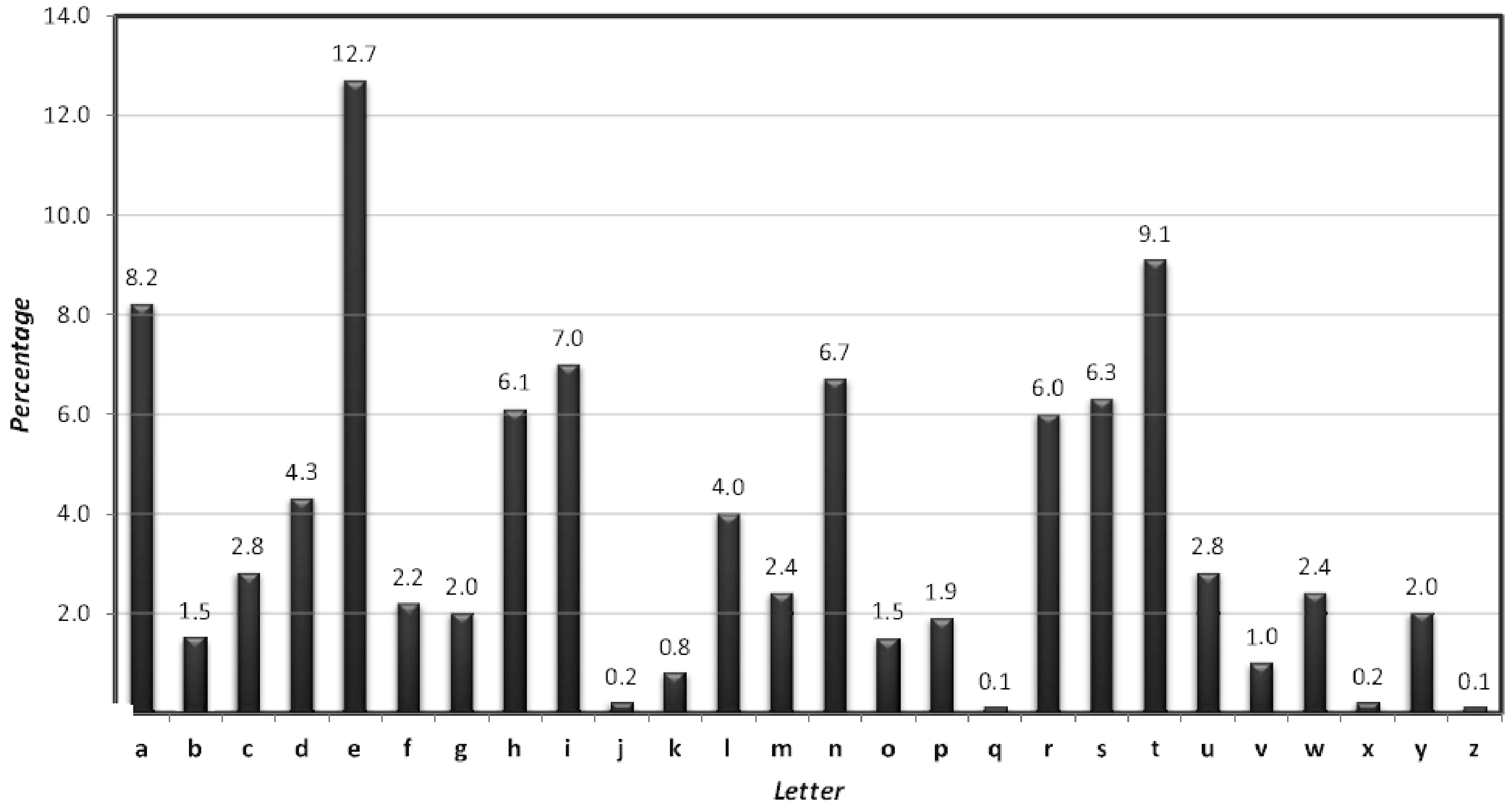
(in which `a` maps to `X`, etc.) would encrypt the message `tellhimaboutme` to `GDOOKVCXEFLGCD`. The name of this cipher comes from the fact that the key defines a (fixed) substitution for individual characters of the plaintext.

Assuming the English alphabet is being used, the key space is of size $26! = 26 \cdot 25 \cdot 24 \cdots 2 \cdot 1$, or approximately 2^{88}, and a brute-force attack is infeasible. This, however, does not mean the cipher is secure! In fact, as we will show next, it is easy to break this scheme even though it has a large key space.

Statistical patterns

- The mono-alphabetic substitution cipher can then be attacked by utilizing statistical patterns of the English language. (Of course, the same attack works for any language.) The attack relies on the facts that:
- For any key, the mapping of each letter is fixed, and so if e is mapped to D, then every appearance of e in the plaintext will result in the appearance of D in the ciphertext.
- The frequency distribution of individual letters in the English language is known. Of course, very short texts may deviate from this distribution, but even texts consisting of only a few sentences tend to have distributions that are very close to the average.

Statistical patterns – Cont'd



An improved attack on the shift cipher

- Try to decipher the following ciphertext—this should convince you how easy the attack is to carry out.

JGRMQOYGHMVBJWRWQFPWHGFFDQGFPFZRKBEEBJIZQQOCIBZKLFAFGQVFZFWWE
OGWOPFGFHWOLPHLRLOLFDMFGQWBLWBWQOLKFWBYLBLYLFSFLJGRMQBOLWJVFP
FWQVHQWFFPQOQVFPQOCFPOGFWFJIGFQVHLHLROQVFGWJVFPFOLFHGQVQVFILE
OGQILHQFQGIQVVOSFAFGBWQVHQWIJVWJVFPFWHGFIWIHZZRQGBABHZQOCGFHX

- It requires decrypting the ciphertext using each possible key, and then checking which key results in a plaintext that “makes sense.”

- A drawback of this approach is that it is somewhat difficult to automate.

An improved attack on the shift cipher – Cont'd

We now describe an attack that does not suffer from these drawbacks. As before, associate the letters of the English alphabet with $0, \ldots, 25$. Let p_i, with $0 \leq p_i \leq 1$, denote the frequency of the ith letter in normal English text (ignoring spaces, punctuation, etc.). Calculation using Figure 1.3 gives

$$\sum_{i=0}^{25} p_i^2 \approx 0.065. \tag{1.1}$$

An improved attack on the shift cipher – Cont'd

Now, say we are given some ciphertext and let q_i denote the frequency of the ith letter of the alphabet in this ciphertext; i.e., q_i is simply the number of occurrences of the ith letter of the alphabet in the ciphertext divided by the length of the ciphertext. If the key is k, then p_i should be roughly equal to q_{i+k} for all i, because the ith letter is mapped to the $(i + k)$th letter. (We use $i+k$ instead of the more cumbersome $[i+k \bmod 26]$.) Thus, if we compute

$$I_j \overset{\text{def}}{=} \sum_{i=0}^{25} p_i \cdot q_{i+j}$$

for each value of $j \in \{0, \ldots, 25\}$, then we expect to find that $I_k \approx 0.065$ (where k is the actual key), whereas I_j for $j \neq k$ will be different from 0.065. This leads to a key-recovery attack that is easy to automate: compute I_j for all j, and then output the value k for which I_k is closest to 0.065.

NYU

Vigenère (poly-alphabetic shift) cipher

- The key defines a mapping that is applied on blocks of plaintext characters.
- Poly-alphabetic substitution ciphers “smooth out” the frequency distribution of characters in the ciphertext and make it harder to perform statistical analysis.
- The *Vigenère cipher*, a special case of the above also called the polyalphabetic shift cipher, works by applying several independent instances of the shift cipher in sequence.
- The key is now viewed as a *string* of letters; encryption is done by shifting each plaintext character by the amount indicated by the next character of the key, wrapping around in the key when necessary.

Plaintext:	`tellhimaboutme`
Key (repeated):	`cafecafecafeca`
Ciphertext:	`VEQPJIREDOZXOE`

Attacking the Vigenère cipher

- Research and find out how attacking the Vigenère cipher works.

Principles of Modern Cryptography

- Principle 1 – Formal Definitions — the first step in solving any cryptographic problem is the formulation of a rigorous and precise definition of security.
  - It should be impossible for an attacker to recover the key.
  - It should be impossible for an attacker to recover the entire plaintext from the ciphertext.
  - It should be impossible for an attacker to recover any character of the plaintext from the ciphertext.
  - Regardless of any information an attacker already has, a ciphertext should leak no additional information about the ununderlying plaintext.

- Principle 2 – Precise Assumptions — when the security of a cryptographic construction relies on an unproven assumption, this assumption must be precisely stated. Furthermore, the assumption should be as minimal as possible.
- Principle 3 – Proofs of Security — cryptographic constructions should be accompanied by a rigorous proof of security with respect to a definition formulated according to principle 1, and relative to an assumption stated as in principle 2 (if an assumption is needed at all).

Threat models

- **Ciphertext-only attack:** This is the most basic attack and refers to a scenario where the adversary just observes a ciphertext (or multiple ciphertexts) and attempts to determine information about the underlying plaintext (or plaintexts). This is the threat model we have been implicitly assuming when discussing classical encryption schemes in the previous section.
- **Known-plaintext attack:** Here, the adversary is able to learn one or more plaintext/ciphertext pairs generated using some key. The aim of the adversary is then to deduce information about the underlying plaintext of some other ciphertext produced using the same key. All the classical encryption schemes we have seen are trivial to break using a known-plaintext attack; we leave a demonstration as an exercise.
- **Chosen-plaintext attack:** In this attack, the adversary can obtain plaintext/ciphertext pairs (as above) for plaintexts of its choice.
- **Chosen-ciphertext attack:** The final type of attack is one where the adversary is additionally able to obtain (some information about) the decryption of ciphertexts of its choice, e.g., whether the decryption of some ciphertext chosen by the attacker yields a valid English message. The adversary's aim, once again, is to learn information about the underlying plaintext of some other ciphertext (whose decryption the adversary is unable to obtain directly).