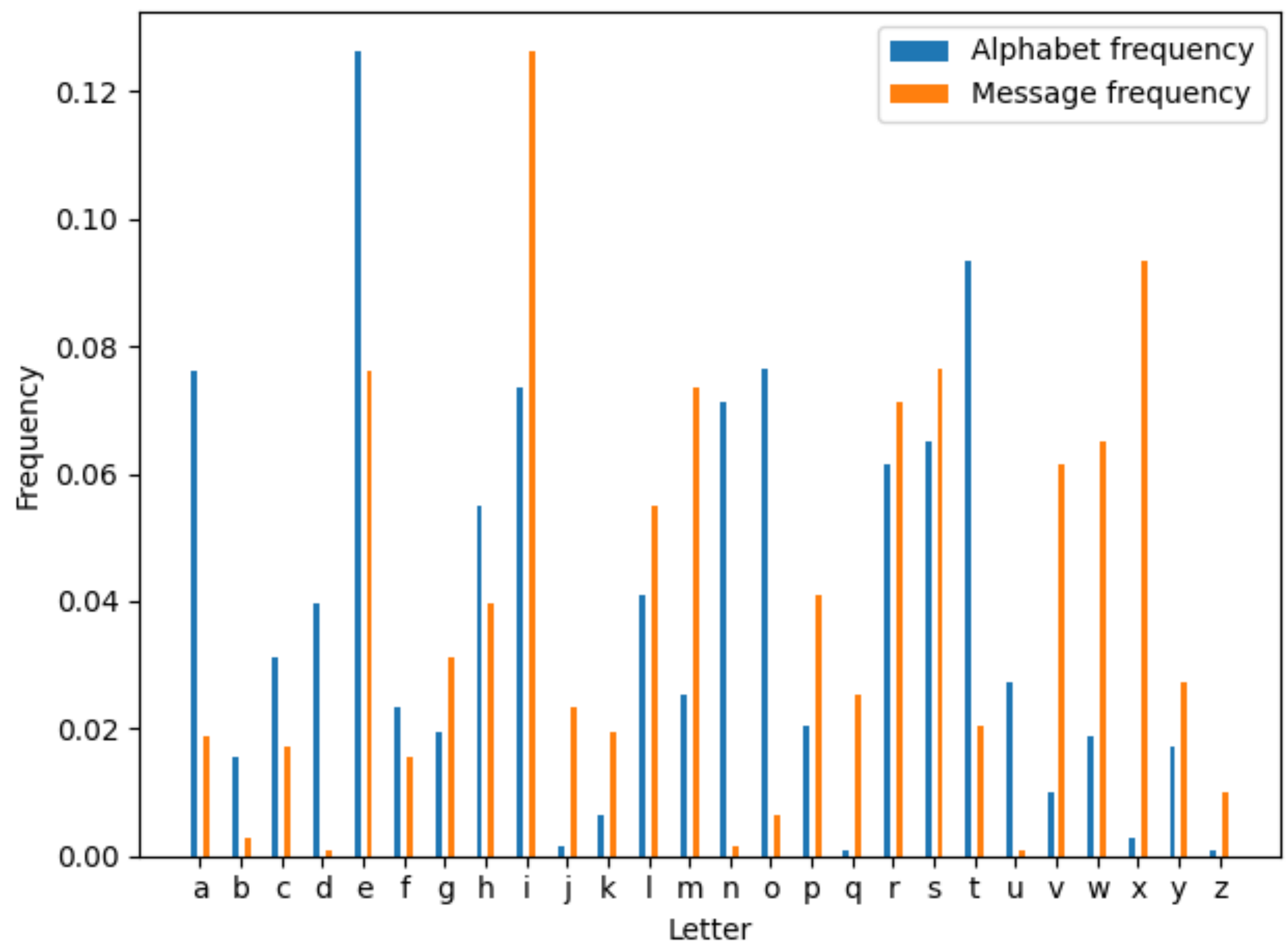
Cryptography - Part 2

Feb. 25, 2025

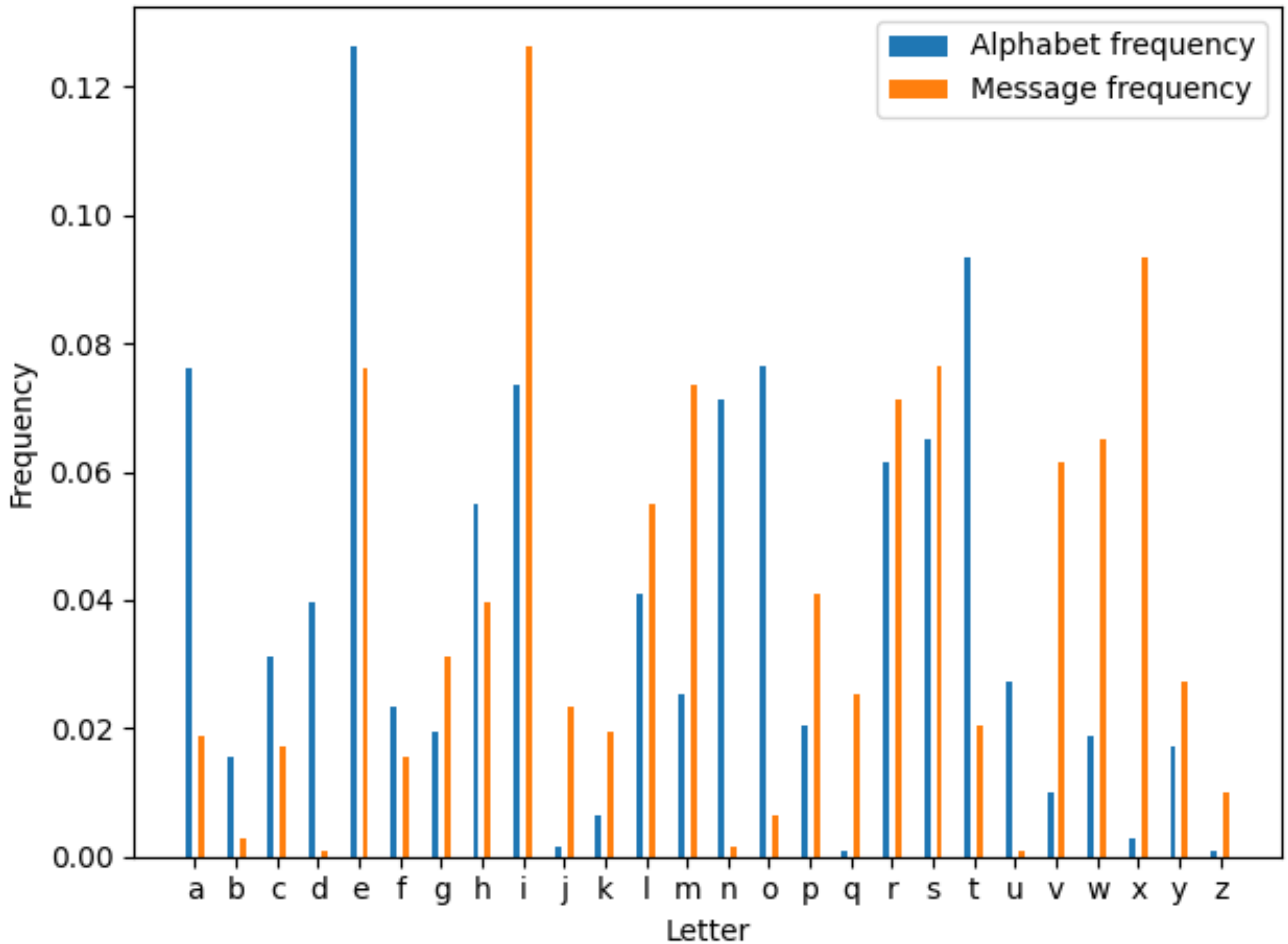
Recap question:

Feb. 25, 2025

A long message gives the frequency analysis in the figure below. The beginning of the message is `xiwx`. Decrypt these first four letters!



The orange bars are just the blue bars shifted to the right 4 steps, i.e., this is a Caesar cipher with shift 4. To decrypt, we shift `xiwx` four steps to the left and obtain `test`.



Cryptography - Part 2

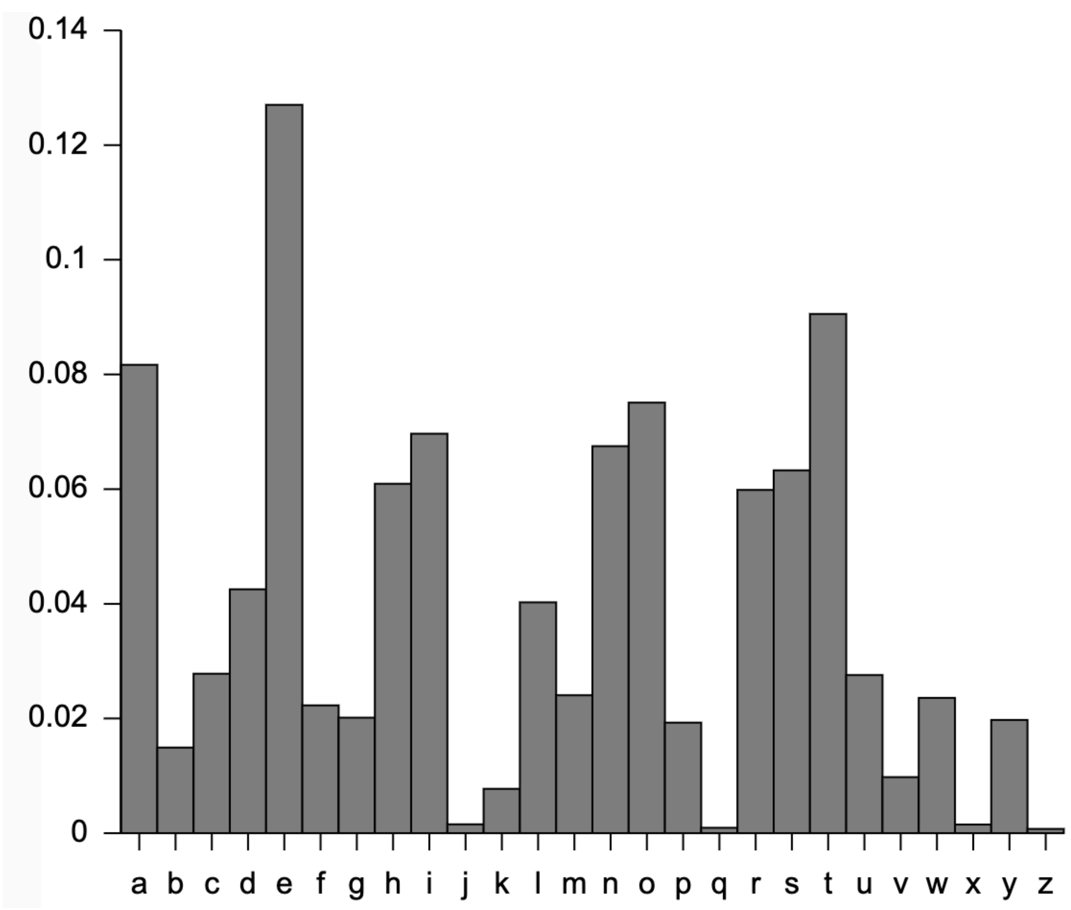
Feb. 25, 2025

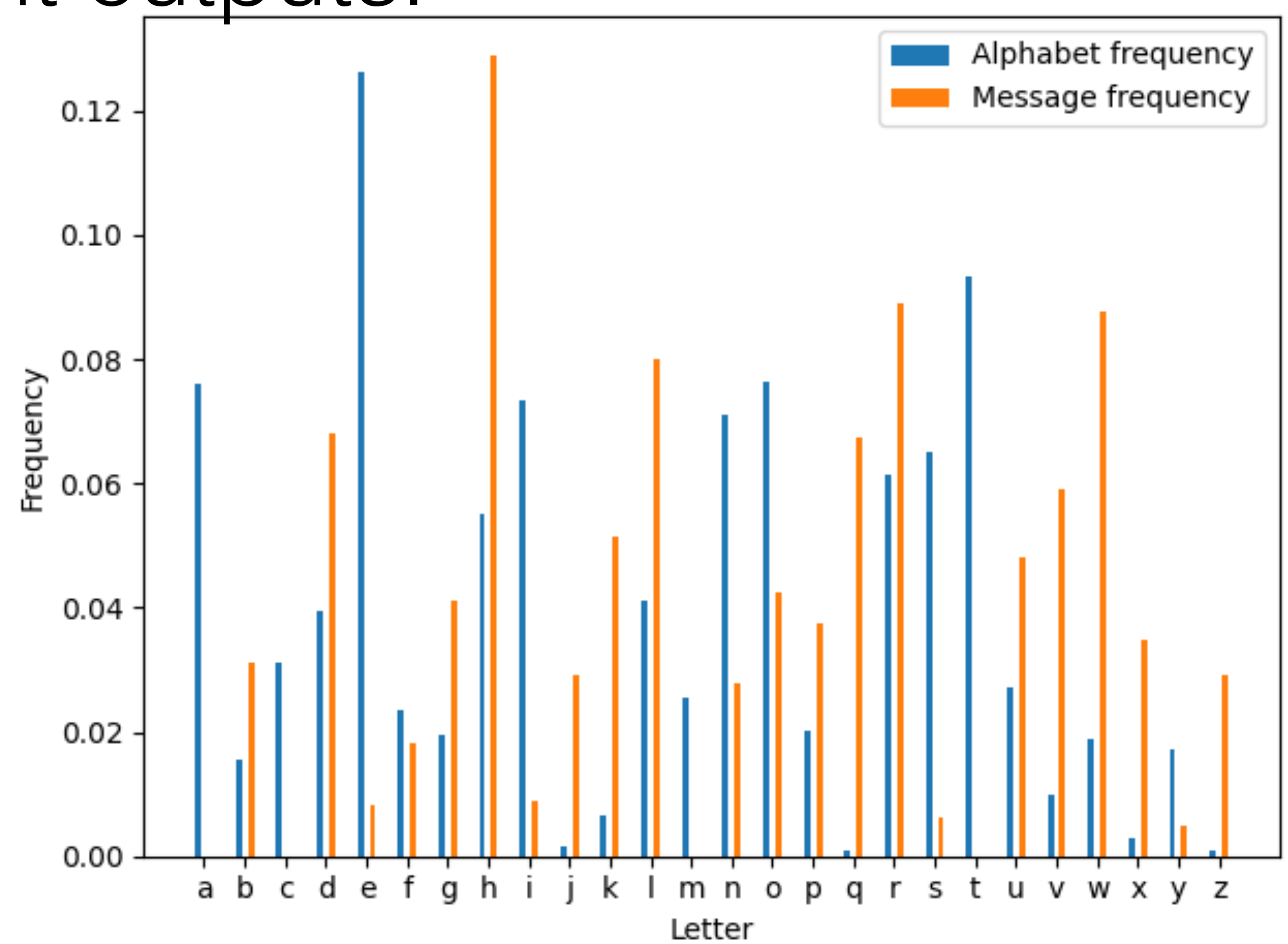
By the end of this lecture, you will be able to:

1. Define one-time pad encryption
2. Perform known-plaintext attacks
3. Explain the encryption used by Hagelin machines

Defeating frequency analysis attacks

To make a coding scheme hard to break, it is important to disguise patterns as much as possible. Typing in the letter ‘M’ twice should for instance give different outputs.





Defeating frequency analysis attacks

To make a coding scheme hard to break, it is important to disguise patterns as much as possible. Typing in the letter ‘M’ twice should for instance give different outputs.

One way to disguise the relative frequency of letters which persists in substitution ciphers is to alternate between different substitution ciphers.

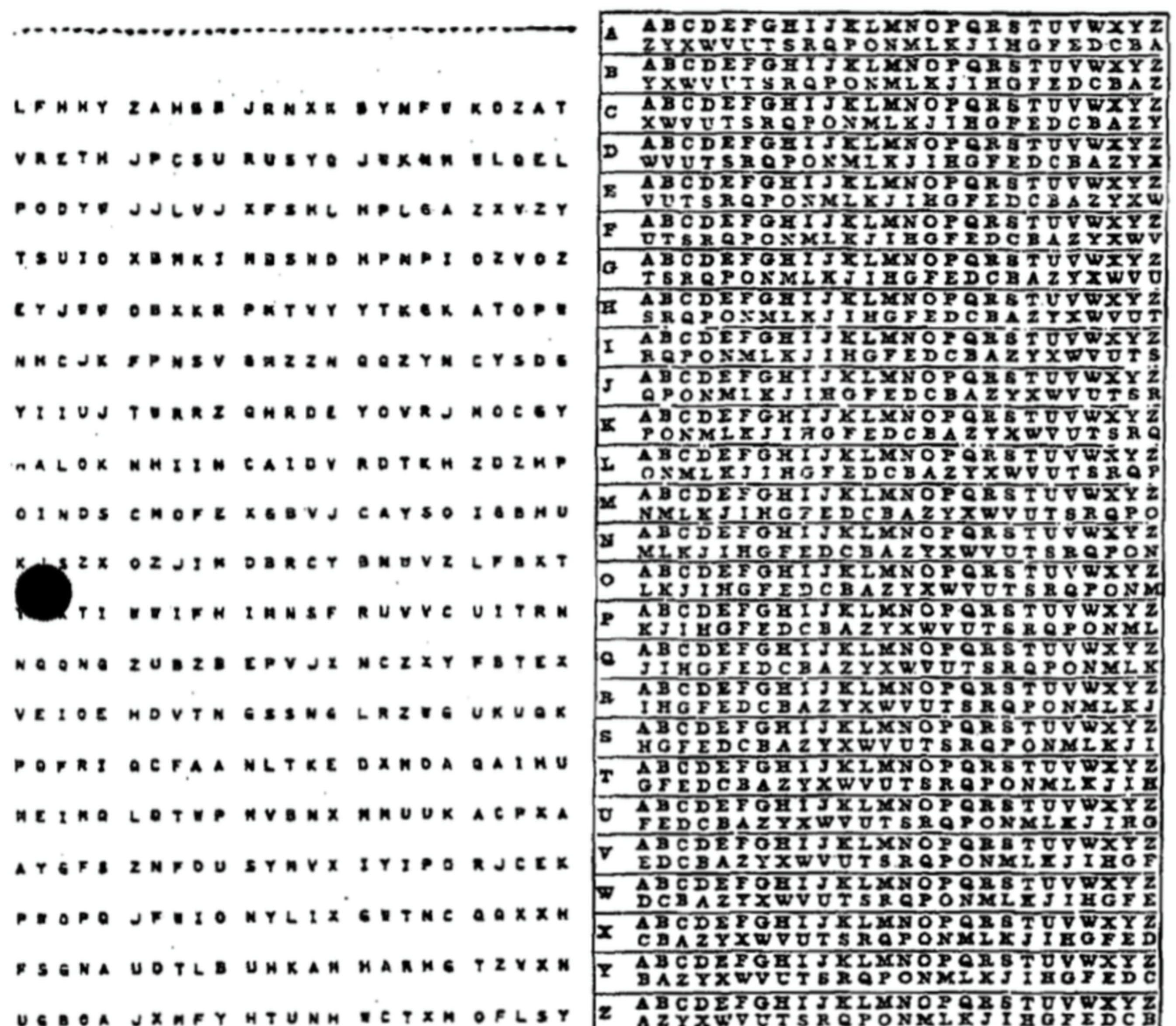
For example, using one scheme for the 1st, 3rd, 5th, ... letters, and another for the 2nd, 4th, 6th, ... letters.

Given a sufficiently long text, the slightly more complex patterns that persist through this scheme are still easy to detect, and every code-breaking expert would make very short work of such an alternating scheme.

The more substitutions used, and the more randomly the different substitutions succeed each other, the harder it becomes to break the code.

Suppose you write a completely random sequence of the numbers 0 to 25, e.g., 23 5 9 0 2 17 21 13 14..., for as many letters as you have in the text you wish to encode, and you use these numbers, one by one, as shifts for the letters in your text.

As long as you never used the same encryption sequence again, your encoded message cannot be decrypted. This approach is called the **one-time pad method.**



As long as you never used the same encryption sequence again, your encoded message cannot be decrypted. This approach is called the **one-time pad method.**

This is unbreakable if the key:

1. is at least as long as the message
2. is generated randomly
3. is kept secret
4. is never reused

Example: Brute-force does not work for one-time pads. Suppose that you intercept the message

```
bwxbeohtmcxt
```

Example: Brute-force does not work for one-time pads. Suppose that you intercept the message

`bwxbeohtmcxt`

You notice that the key 1,3,4,1,2,4,7,0,9,2,1,6 decodes the message to

`attackatdawn`

Example: Brute-force does not work for one-time pads. Suppose that you intercept the message

`bwxbeohtmcxt`

You notice that the key 1,3,4,1,2,4,7,0,9,2,1,6 decodes the message to

`attackatdawn`

Your friend notices that the key 1,3,4,1,2,4,7,0,25,14,9,6 decodes the message to

`attackatnoon`

Example: Brute-force does not work for one-time pads. Suppose that you intercept the message

`bwxbeohtmcxt`

You notice that the key 1,3,4,1,2,4,7,0,9,2,1,6 decodes the message to

`attackatdawn`

Your friend notices that the key 1,3,4,1,2,4,7,0,25,14,9,6 decodes the message to

`attackatnoon`

Your other friend notices that the key 19,15,23,6,0,21,11,5,10,2,4,1 decodes the message to

`ihavetwocats`

Does one-time pad encryption solve every problem?

Does one-time pad encryption solve every problem?

1. Using a new key for every message (and every person) is cumbersome.
2. The sender and receiver need to agree on keys beforehand and cannot e.g. send new keys unencrypted over the internet.
3. It is not entirely easy to generate random keys
4. …

Why is reusing the one-time pad a bad idea?

Reason 1: Frequency analysis could then be carried out for each letter.

Why is reusing the one-time pad a bad idea?

Reason 1: Frequency analysis could then be carried out for each letter.

<u>Message</u>		<u>Encrypted text</u>
hello	$\longrightarrow$	ihpmq
truck	$\longrightarrow$	uuydm
malls	$\longrightarrow$	ndpmu
$\vdots$		$\vdots$

Why is reusing the one-time pad a bad idea?

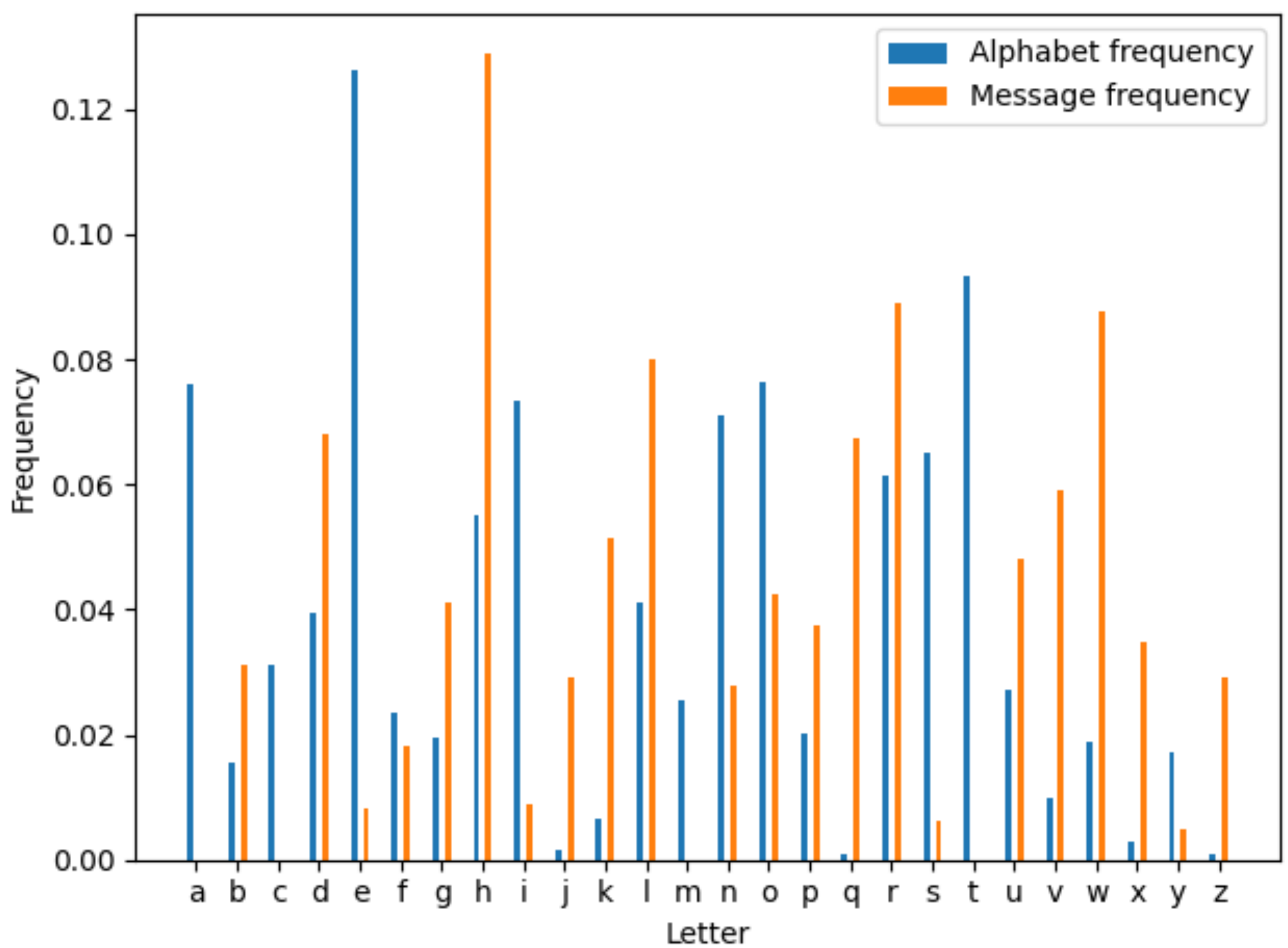
Reason 1: Frequency analysis could then be carried out for each letter.

<u>Message</u>		<u>Encrypted text</u>
hello	$\longrightarrow$	ihpmq
truck	$\longrightarrow$	uuydm
malls	$\longrightarrow$	ndpmu
$\vdots$		$\vdots$

Why is reusing the one-time pad a bad idea?

Reason 1: Frequency analysis could then be carried out for each letter.

Message	Encrypted text
hello →	**i**hpmq
→	**u**uydm
→	**n**dpmu
	⋮



Why is reusing the one-time pad a bad idea?

Reason 1: Frequency analysis could then be carried out for each letter.

Message		Encrypted text
hello	→	i**h**pmq
truck	→	u**u**ydm
malls	→	n**d**pmu
⋮		⋮

Why is reusing the one-time pad a bad idea?

Reason 1: Frequency analysis could then be carried out for each letter.

<u>Message</u>		<u>Encrypted text</u>
hello	⟶	i**h**pmq
truck	⟶	u**u**ydm
malls	⟶	n**d**pmu
⋮		⋮

Continue until we have the entire key.

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

Assume that we intercept two messages, encrypted with the same one-time pad. Also assume that we know that the last 6 letters of message 1 are "london". Find the last 6 digits of the key and use this to decrypt the last 6 letters of message 2. See if you can then guess the entire message 2.

Message 1 $\longrightarrow$ `crqcnsudxp`

Message 2 $\longrightarrow$ `bwxbeoyovg`

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

* * * * london	$\longrightarrow$	crqcnsudxp
* * * * * * * * * *	$\longrightarrow$	bwxbeoyovg

Key: * * * * * * * * * *

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

****london $\longrightarrow$ crqcnsudxp

********** $\longrightarrow$ bwxbeoyovg

Key: ****2*****

$\xrightarrow{2}$

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

`****london` ⟶ `crqcnsudxp`

`**********` ⟶ `bwxbeoyovg`

Key: `****24****`

$\xrightarrow{4}$

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

`****london` ⟶ `crqcnsudxp`

`**********` ⟶ `bwxbeoyovg`

Key: `****247***`

7

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

****london → crqcnsudxp

********** → bwxbeoyovg

Key: ****2470**

0

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

*****london $\longrightarrow$ crqcnsudxp

*********** $\longrightarrow$ bwxbeoyovg

Key: *****24709*

$$\xrightarrow{\quad 9 \quad}$$

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

****london $\longrightarrow$ crqcnsudxp

********** $\longrightarrow$ bwxbeoyovg

Key: ****247092

$\xrightarrow{2}$

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

`****london`	→	`crqcnsudxp`
`****c*****`	→	`bwxbeoyovg`

Key: `****247092`

2

←

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

****london → crqcnsudxp

****ck**** → bwxbeoyovg

Key: ****247092

4

←

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

`****london` → `crqcnsudxp`

`****ckr***` → `bwxbeoyovg`

Key: `****247092`

$$\xleftarrow{\quad 7 \quad}$$

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

****london ⟶ crqcnsudxp

****ckr**o**** ⟶ bwxbeoy**o**vg

Key: ****247**0**92

0

⟵

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

****london ⟶ crqcnsudxp

****ckro**m*** ⟶ bwxbeoyo**v**g

Key: ****2470**9**2

9

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

```
****london    ──────►    crqcnsudxp
****ckrome    ──────►    bwxbbeoyovg
```

Key: ****247092

2 ←

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

Why is reusing the one-time pad a bad idea?

Reason 2: Given two messages, part of the key can sometimes be recovered using **known-plaintext attacks.**

`****london`	$\longrightarrow$	`crqcnsudxp`
`****ckrome`	$\longrightarrow$	`bwxbeoyovg`

Key: `****247092`

From the context, the code breaker might guess that the entire message 2 then is “attack Rome”. From the first 4 letters, we can then recover the first 4 letters of the key, in the same way, and then decrypt the entire message 1 to “bomb London”.

Why is reusing the one-time pad a bad idea?

What if we do not know where the word “London” appears in message 1 (just that it appears somewhere)?

Why is reusing the one-time pad a bad idea?

What if we do not know where the word “London” appears in message 1 (just that it appears somewhere)? Try all places and see if the procedure gives legible text for message 2!

Why is reusing the one-time pad a bad idea?

What if we do not know where the word "London" appears in message 1 (just that it appears somewhere)? Try all places and see if the procedure gives legible text for message 2!

london**** $\longrightarrow$ crqcnsudxp

ktucfj**** $\longrightarrow$ bwxbeoyovg

Key: 17,3,3,25,25,5,****

Why is reusing the one-time pad a bad idea?

What if we do not know where the word "London" appears in message 1 (just that it appears somewhere)? Try all places and see if the procedure gives legible text for message 2!

*london*** ⟶ crqcnsudxp

*qvmukr*** ⟶ bwxbeoyovg

Key: *6,2,15,10,4,7,***

Why is reusing the one-time pad a bad idea?

What if we do not know where the word "London" appears in message 1 (just that it appears somewhere)? Try all places and see if the procedure gives legible text for message 2!

$$**\text{london}** \longrightarrow \text{crqcnsudxp}$$

$$**\text{snezsy}** \longrightarrow \text{bwxbeoyovg}$$

Key: **5,14,0,15,6,16,**

Why is reusing the one-time pad a bad idea?

What if we do not know where the word "London" appears in message 1 (just that it appears somewhere)? Try all places and see if the procedure gives legible text for message 2!

***london*	$\longrightarrow$	crqcnsudxp
***kfjhzl*	$\longrightarrow$	bwxbeoyovg

Key: ***17,25,5,17,15,10,*

Why is reusing the one-time pad a bad idea?

What if we do not know where the word "London" appears in message 1 (just that it appears somewhere)? Try all places and see if the procedure gives legible text for message 2!

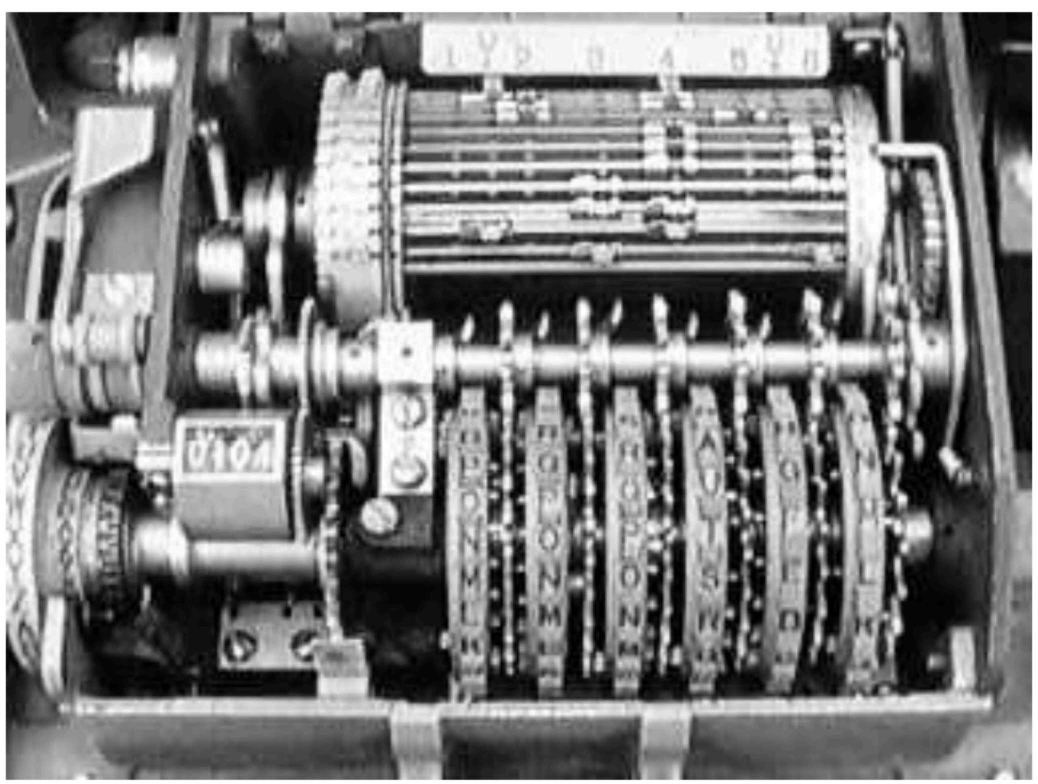
****london	$\longrightarrow$	crqcnsudxp
****ckrome	$\longrightarrow$	bwxbeoyovg

Key: ****2,4,7,0,9,2

Only the last position gives legible text in message 2, so we have found the correct position of "London"

Hagelin machines

Idea: make a machine that goes from the substitution cipher of one letter to the substitution cipher of the next in a way that is complicated and seems random

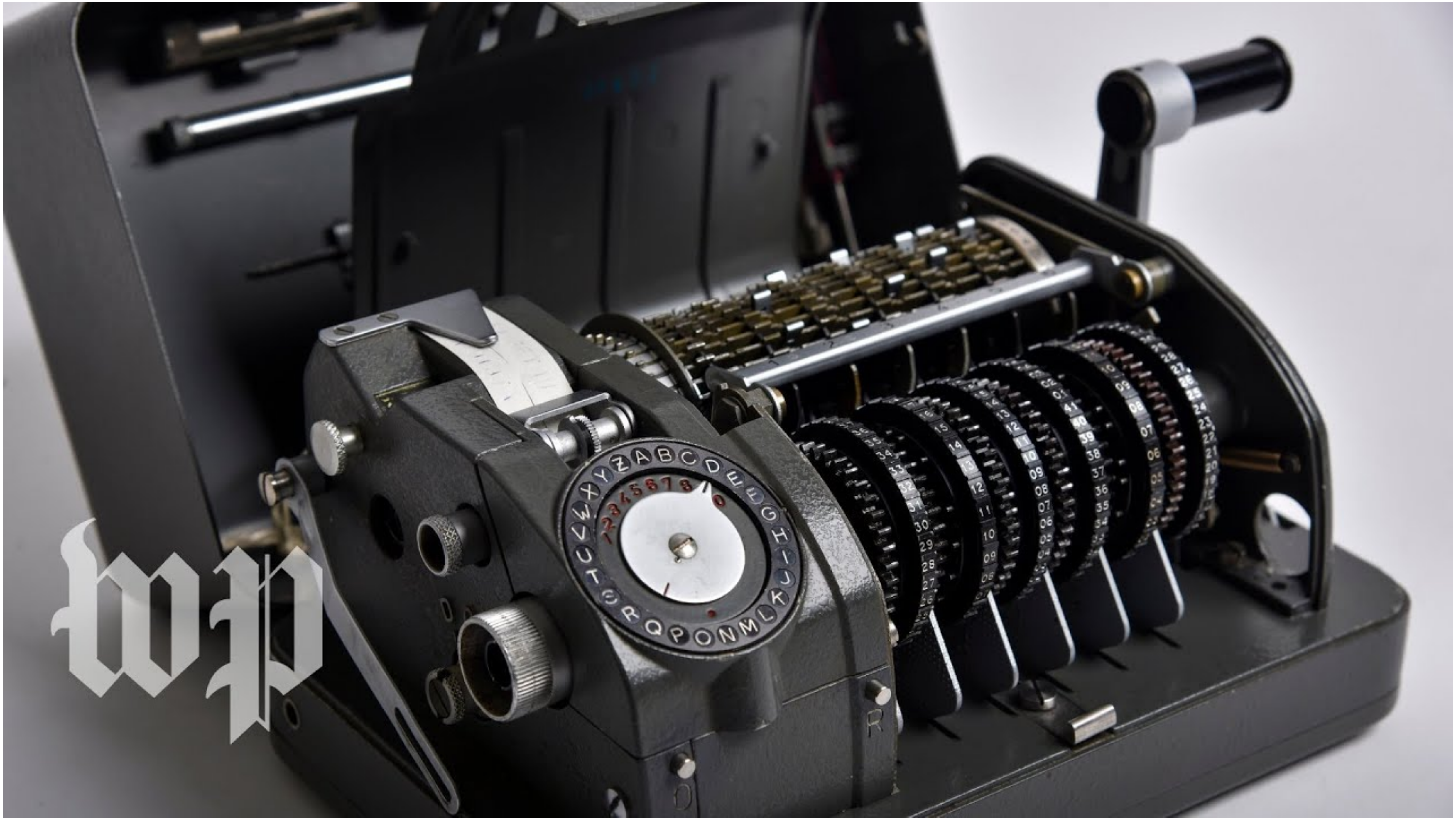


Hagelin machine
(1940s - 1950s)

This machine uses ever-changing “mirrored Caesar codes”: for each letter, in the message to be encoded, the machine first determines a shift of the alphabet, and then writes the shifted alphabet in reverse order.

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
F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G

The Hagelin machine changes the shift for every letter to be encoded via an ingenious system that tries to make the successive shifts very “random”.



https://www.youtube.com/watch?v=V6ev1xL1TPs

The six rotor gears from left to right have respectively 26, 25, 23, 21, 19, and 17 teeth. The leftmost rotor will therefore be in its original position after every 26 cranks, the second after every 25 cranks, and so on.

One can then ask after how many cranks (N) the rotors will all be back in their original position; N determines the period of the whole encryption scheme, i.e., the number of encryptions after which the pattern starts repeating.

$$
\begin{aligned}
N &= 26 \times 25 \times 23 \times 21 \times 19 \times 17 \\
&= 101,405,850
\end{aligned}
$$

However, even though it takes this very large number of cranks before the whole system repeats exactly, because it is made up of six rotors, each with its own system of pins that cycle back to their original position much faster, the whole mechanism leads to more patterns than a single rotating wheel with N gears would have.

In addition, if the machine, with the same settings, is used by many senders (e.g. in a military setting at war), then the code-breaker can use patterns present in many of their messages.

As a result, this code can be (and was) broken, especially if it is used unwisely, for instance, if the internal settings, governed by the pins and lugs, are not changed sufficiently frequently.

Enigma machine

The Enigma machine functions in a similar way. Each day, each operator of the machines around the world used one pre-determined setting for the rotors.

If the Allies could decrypt one message a day, they could then deduce the key and read all other messages.

Enigma machine

Part of the strategy was to use plaintext attacks. Which words did they test?

1. “Weather” (and the actual weather)

Enigma machine

Part of the strategy was to use plaintext attacks. Which words did they test?

1. “Weather” (and the actual weather)
2. “Nothing to report”

Enigma machine

Part of the strategy was to use plaintext attacks. Which words did they test?

1. “Weather” (and the actual weather)
2. “Nothing to report”
3. Digits (these were written with letters)

Enigma machine

Part of the strategy was to use plaintext attacks. Which words did they test?

1. “Weather” (and the actual weather)
2. “Nothing to report”
3. Digits (these were written with letters)
4. “Mines” (after dropping mines in areas surveilled by the Germans)

Enigma machine

Part of the strategy was to use plaintext attacks. Which words did they test?

1. “Weather” (and the actual weather)
2. “Nothing to report”
3. Digits (these were written with letters)
4. “Mines” (after dropping mines in areas surveilled by the Germans)
5. Longer messages passed on by double-agents to their handlers
6. …

Enigma machine

Part of the strategy was to use plaintext attacks. Which words did they test?

1. “Weather” (and the actual weather)
2. “Nothing to report”
3. Digits (these were written with letters)
4. “Mines” (after dropping mines in areas surveilled by the Germans)
5. Longer messages passed on by double-agents to their handlers

This narrowed down the possible choices of the rotors enough that decryption could be finished by other techniques

How to generate random numbers?

How to generate random numbers?

1. Use the Hagelin machine?
2. Throw dice (or similar)?
3. Ask people to come up with a number between 0 and 25?
4. Use a computer?

How to generate random numbers?

1. Use the Hagelin machine?
2. Throw dice (or similar)?
3. Ask people to come up with a number between 0 and 25?
4. Use a computer?

Pseudorandom number generator (PNG):

an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.