Sending Secrets:
Cryptography and Privacy in a Quantum World

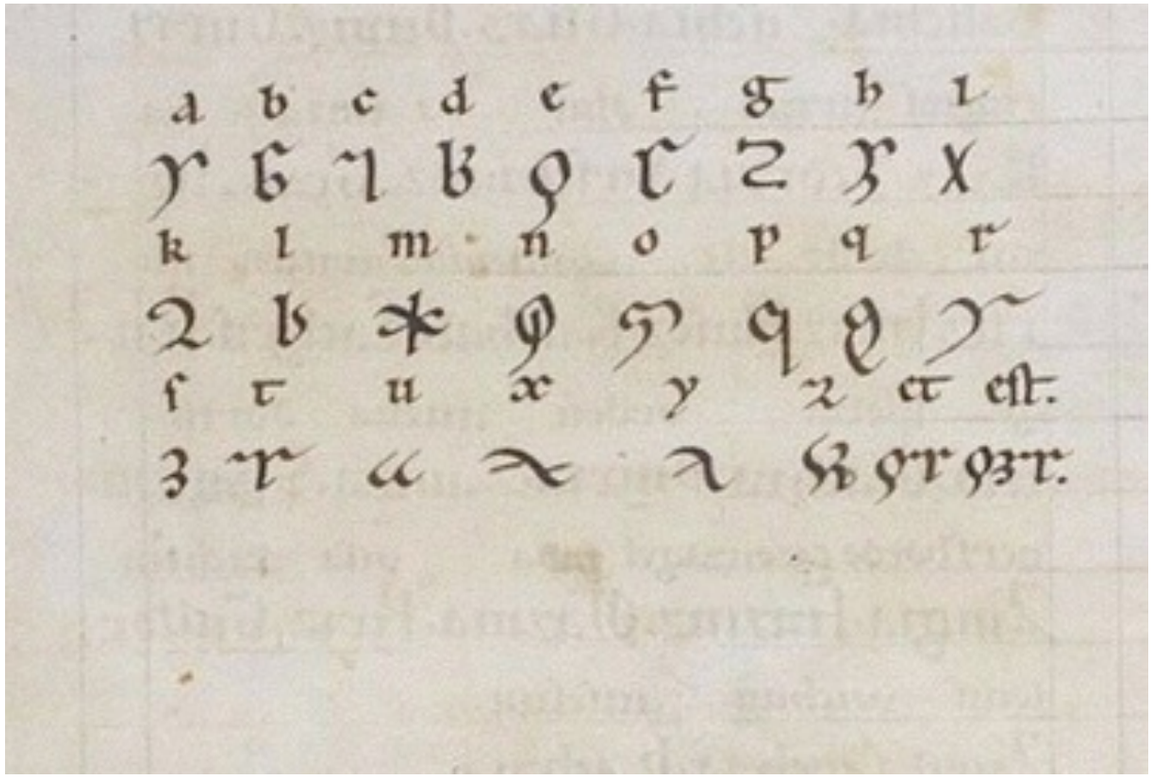
Cristopher Moore
Santa Fe Institute

Harpers, issue 179, June/November 1939

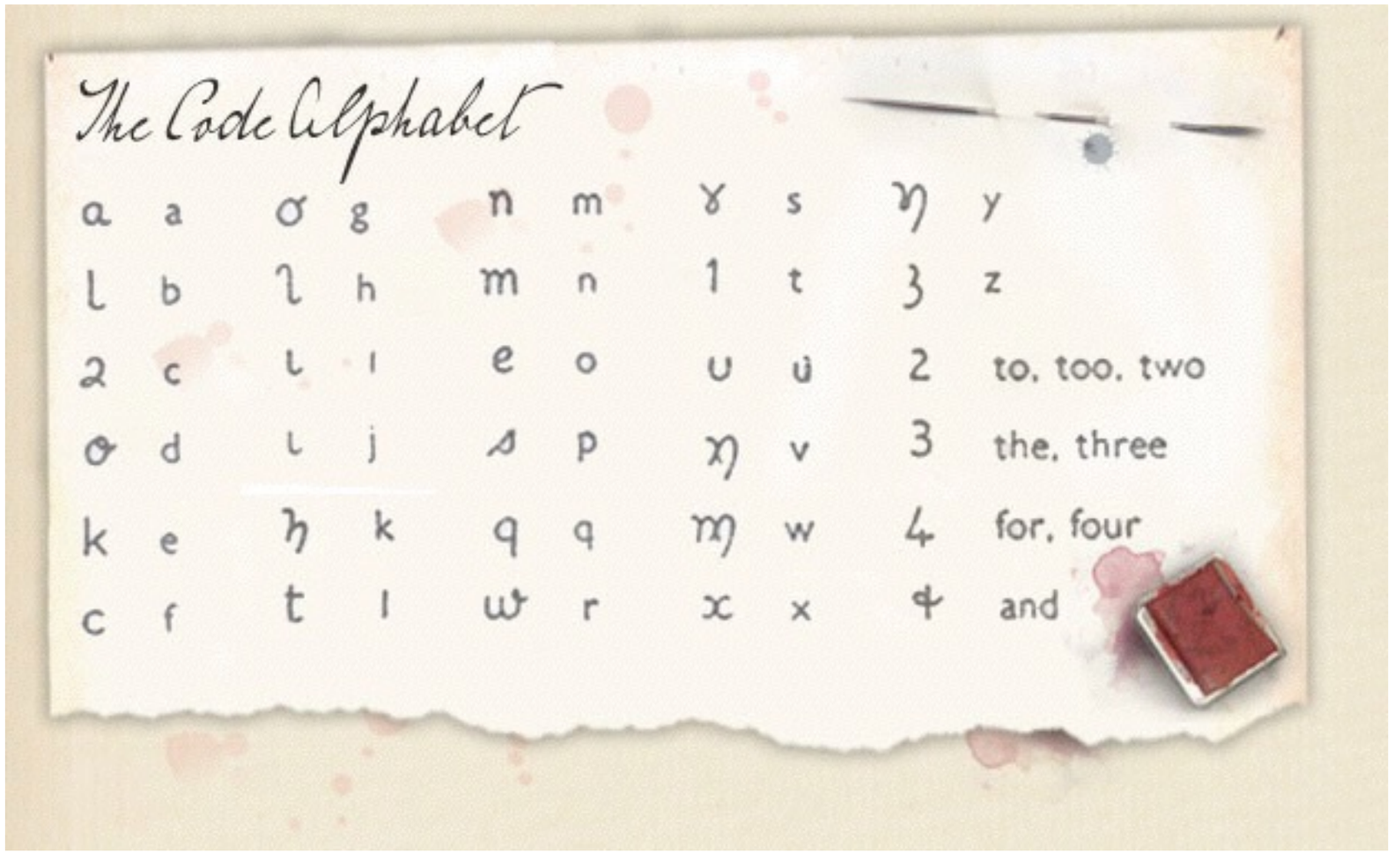
Is it not a curious fact that in a world steeped in irrational hatreds which threaten civilization itself, men and women—old and young—detach themselves wholly or partly from the angry current of daily life to devote themselves to the cultivation of beauty, to the extension of knowledge, to the cure of disease, to the amelioration of suffering, just as though fanatics were not simultaneously engaged in spreading pain, ugliness, and suffering?

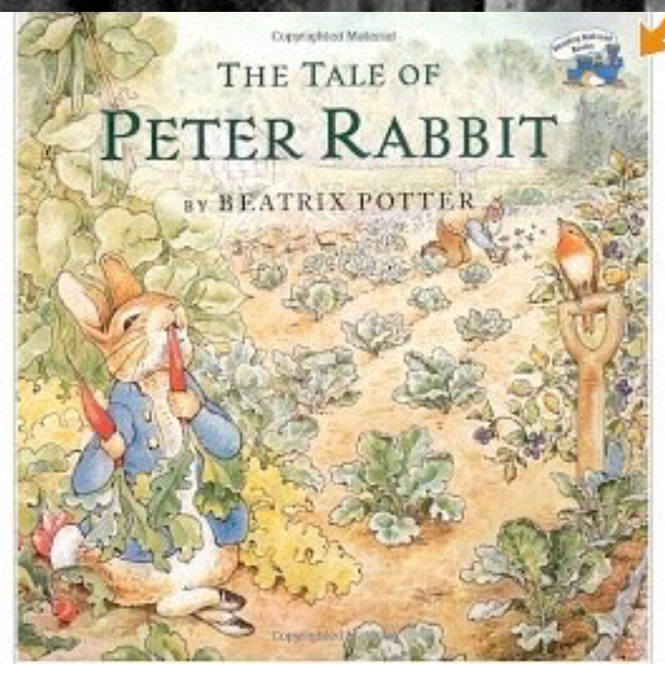
Writing

Secret writing

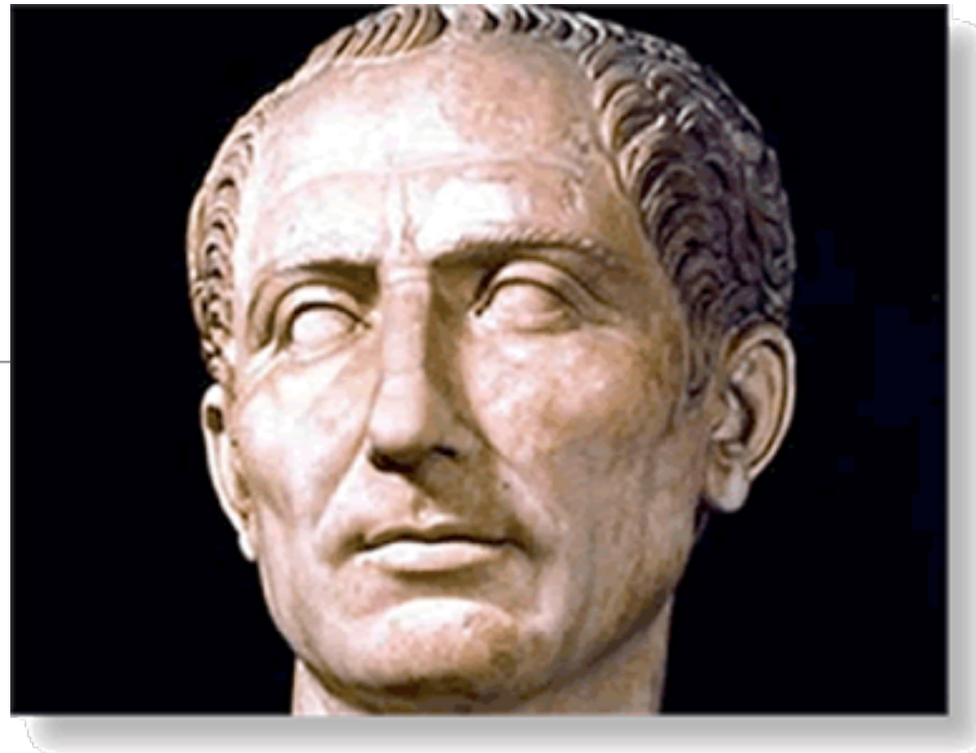


Secret writing

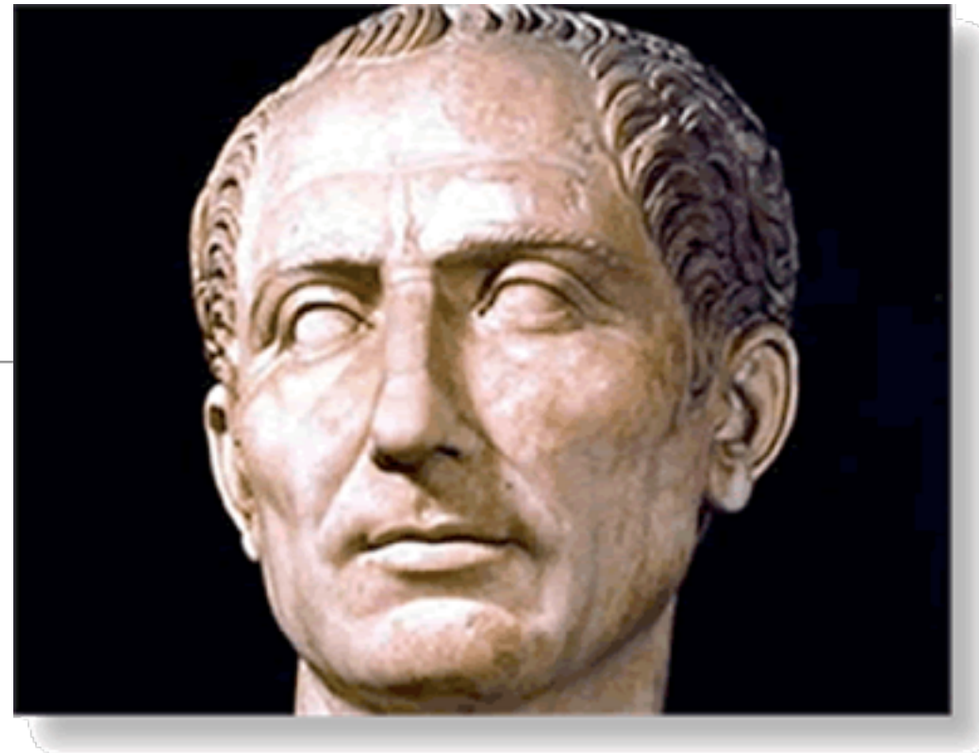




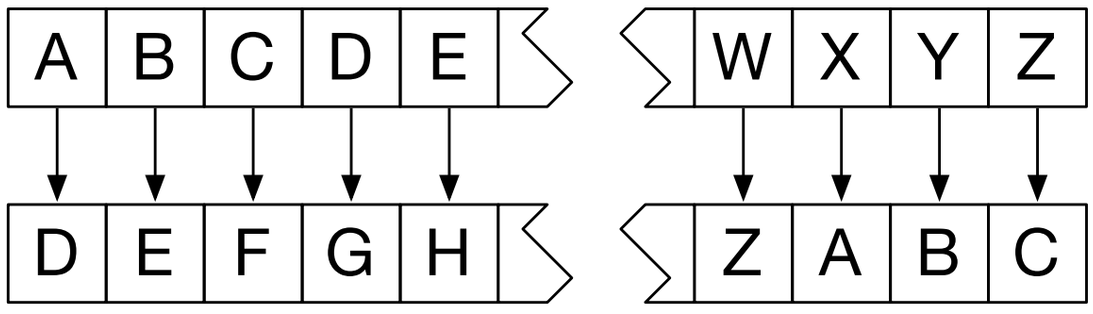
Caesar’s cipher



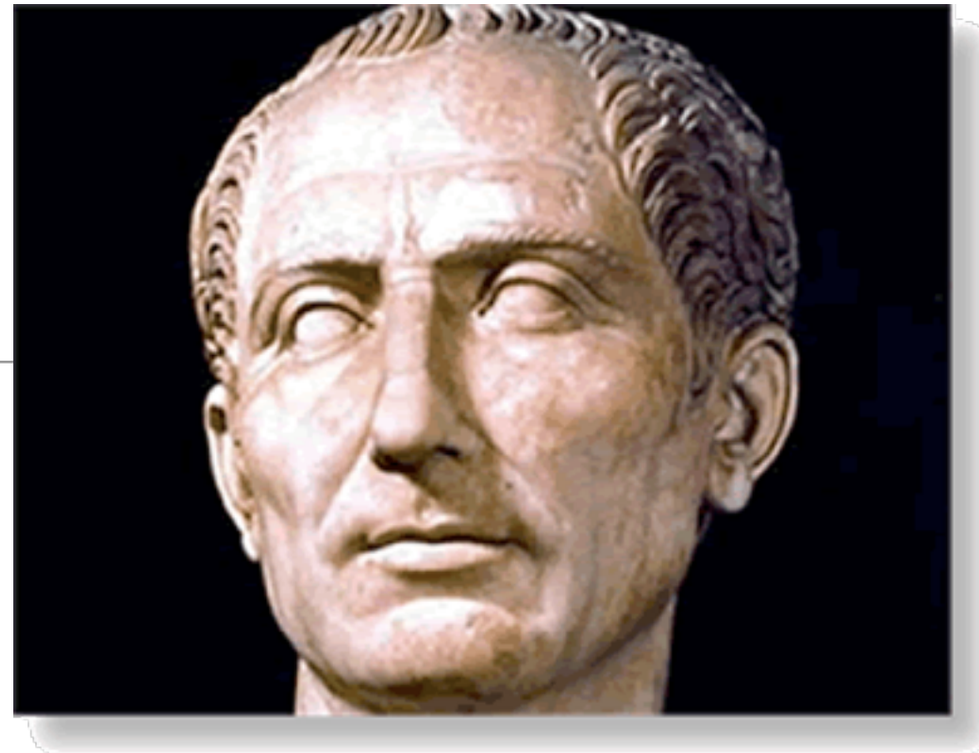
Caesar’s cipher



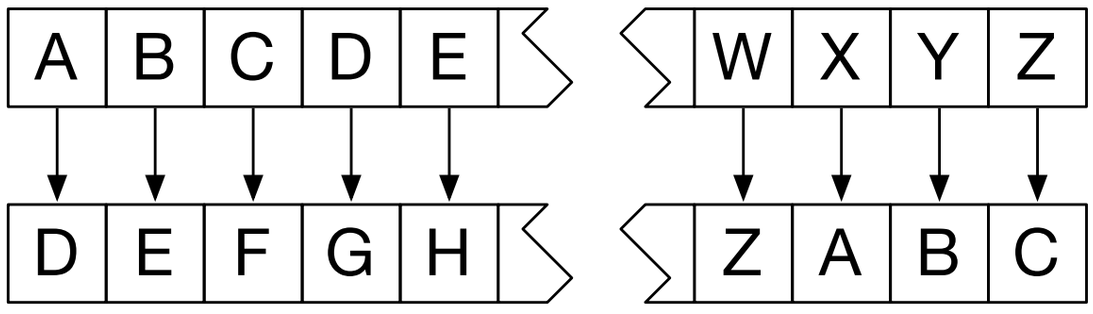
Shift each letter three places in the alphabet:



Caesar's cipher

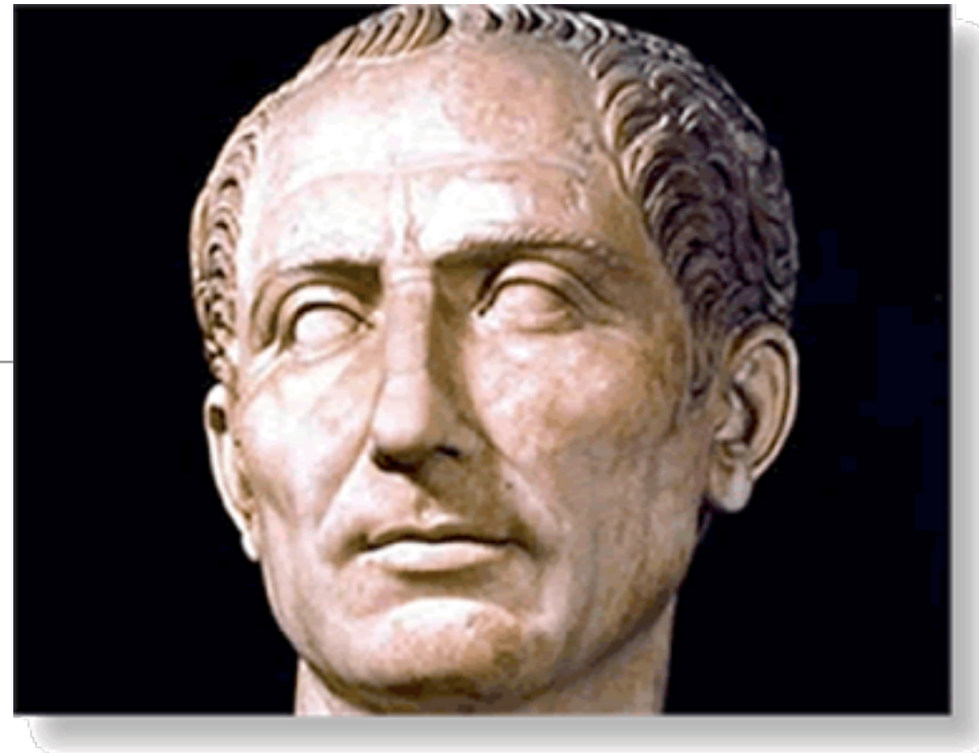


Shift each letter three places in the alphabet:

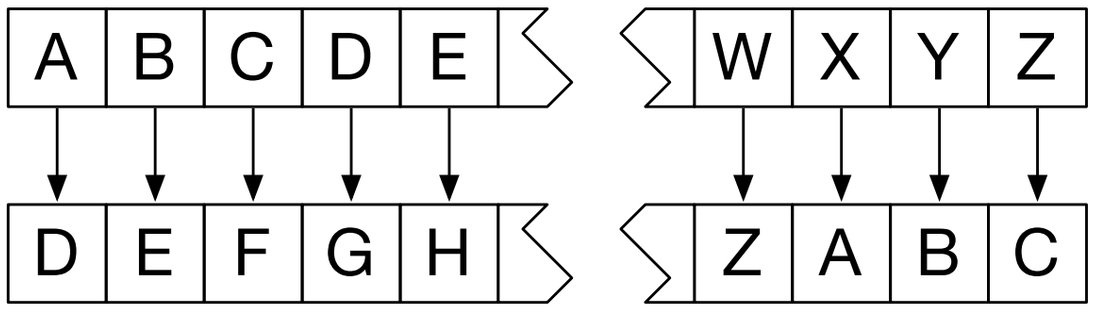


Plaintext: `ET TU BRUTE`

Caesar's cipher



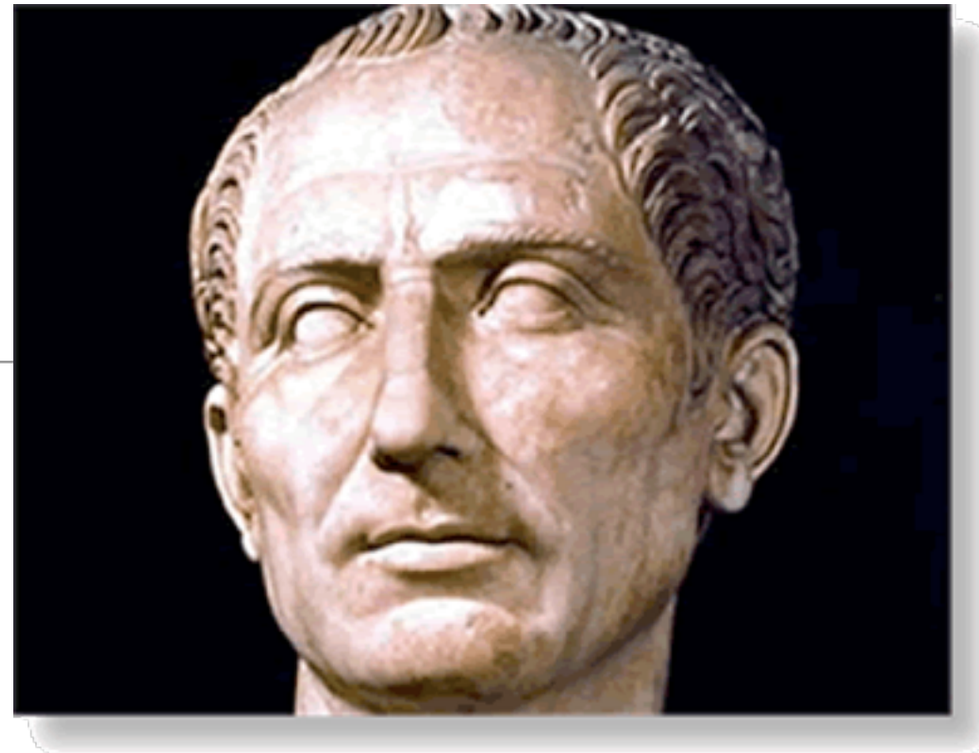
Shift each letter three places in the alphabet:



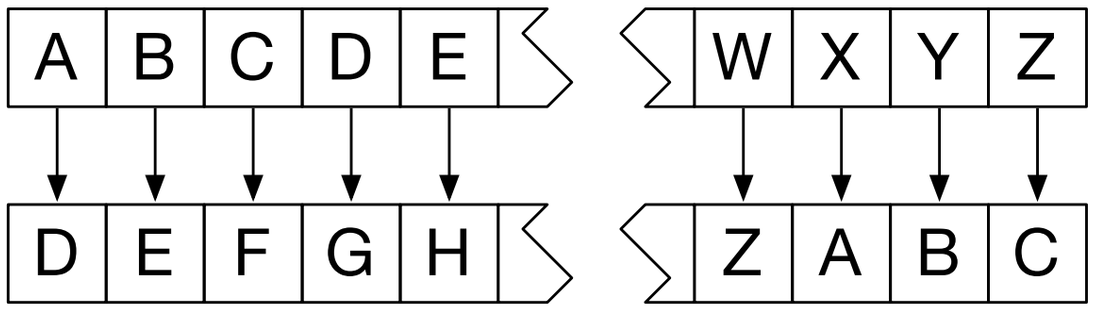
Plaintext: `ET TU BRUTE`

Ciphertext: `HW WX EUXWH`

Caesar's cipher



Shift each letter three places in the alphabet:

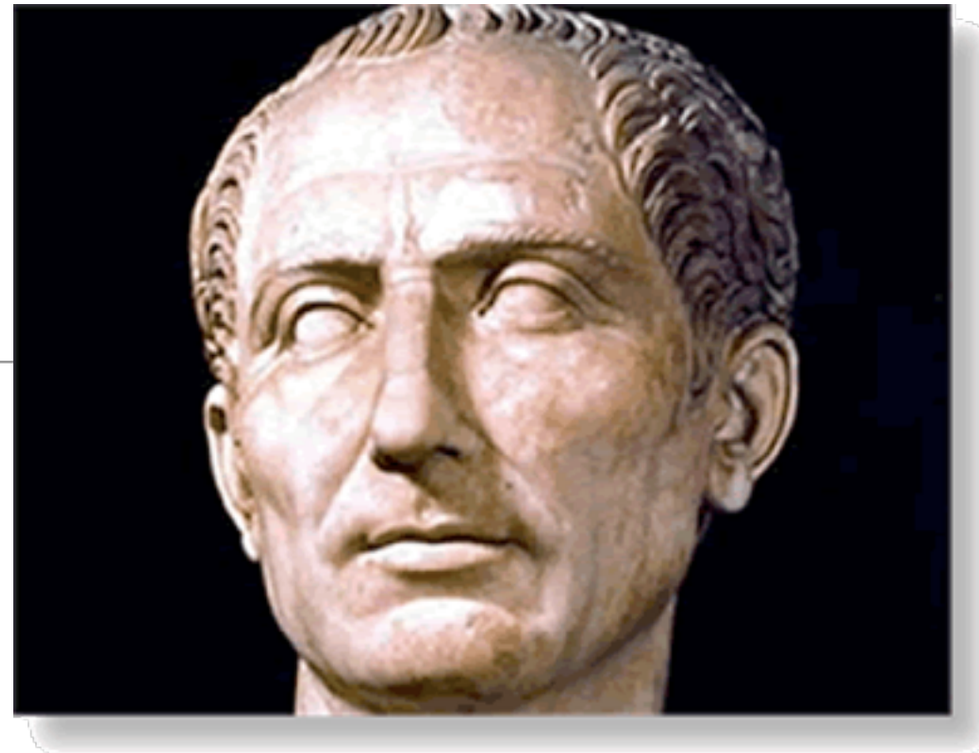


Plaintext: `ET TU BRUTE`

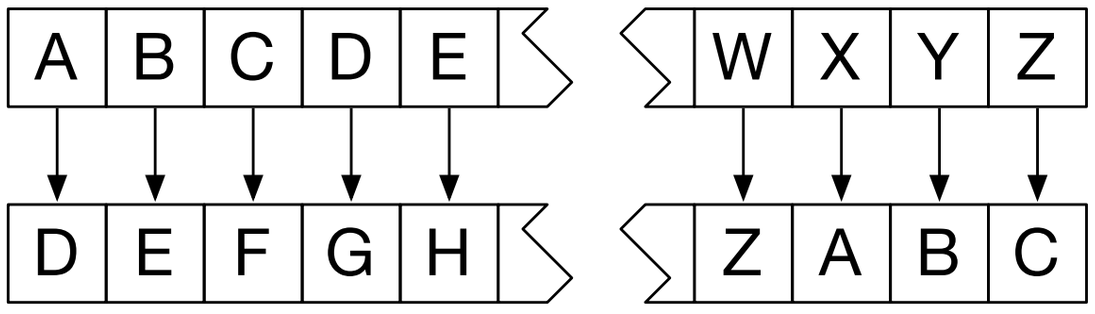
Ciphertext: `HW WX EUXWH`

To decode, just shift backwards

Caesar's cipher



Shift each letter three places in the alphabet:



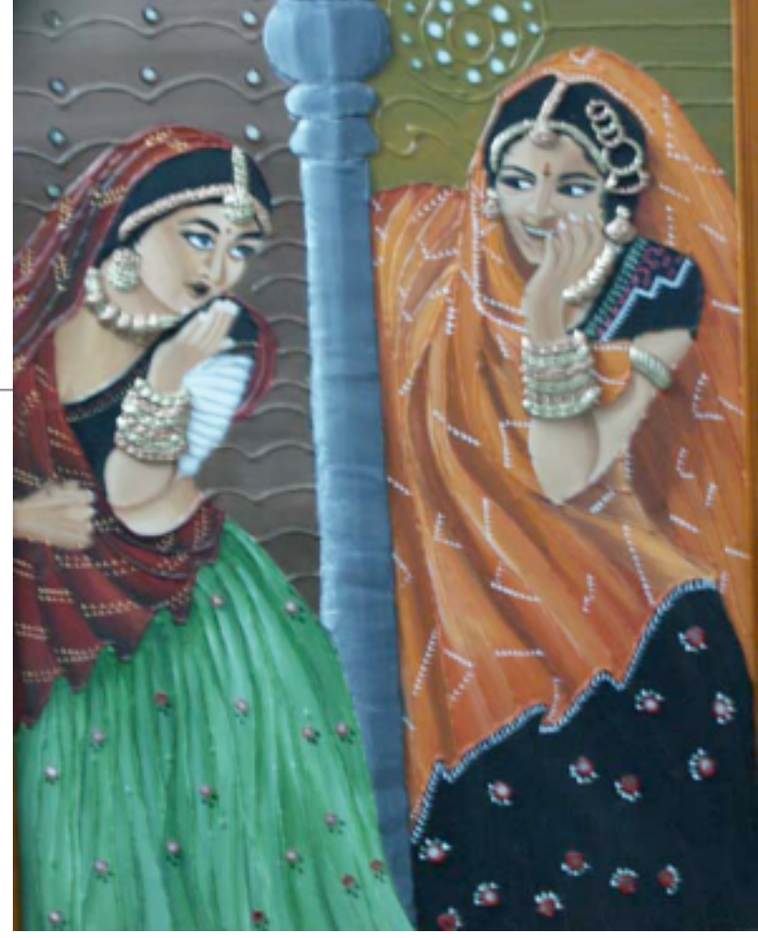
Plaintext: `ET TU BRUTE`

Ciphertext: `HW WX EUXWH`

To decode, just shift backwards

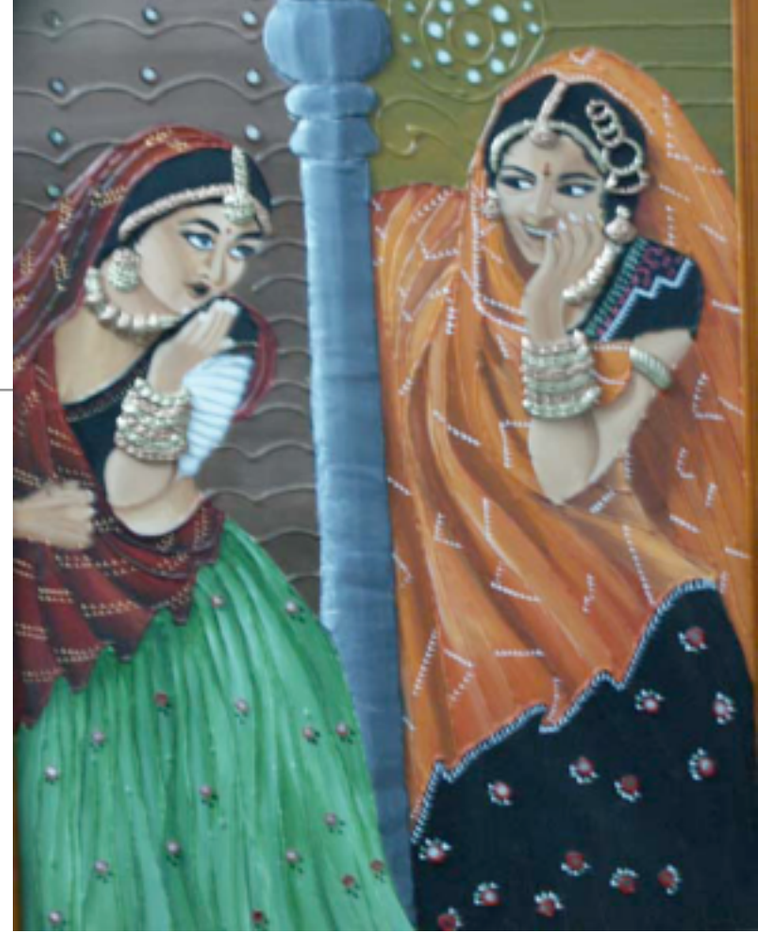
26 possibilities

Art number 45



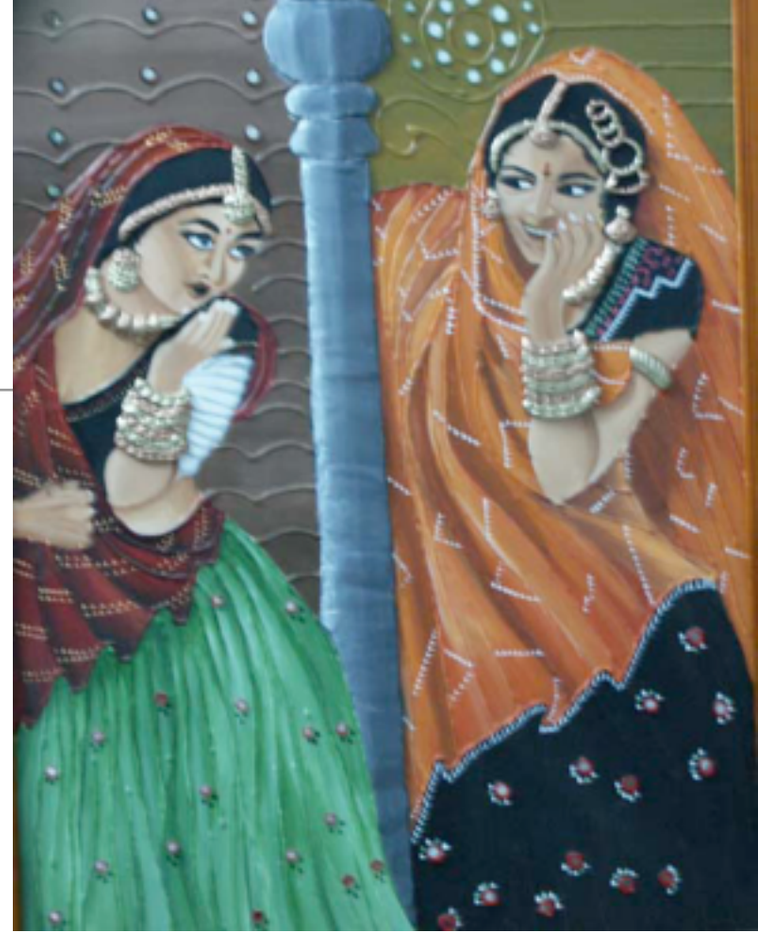
Art number 45

The Kama Sutra recommends that women learn the art of secret writing, to help them conceal their liaisons

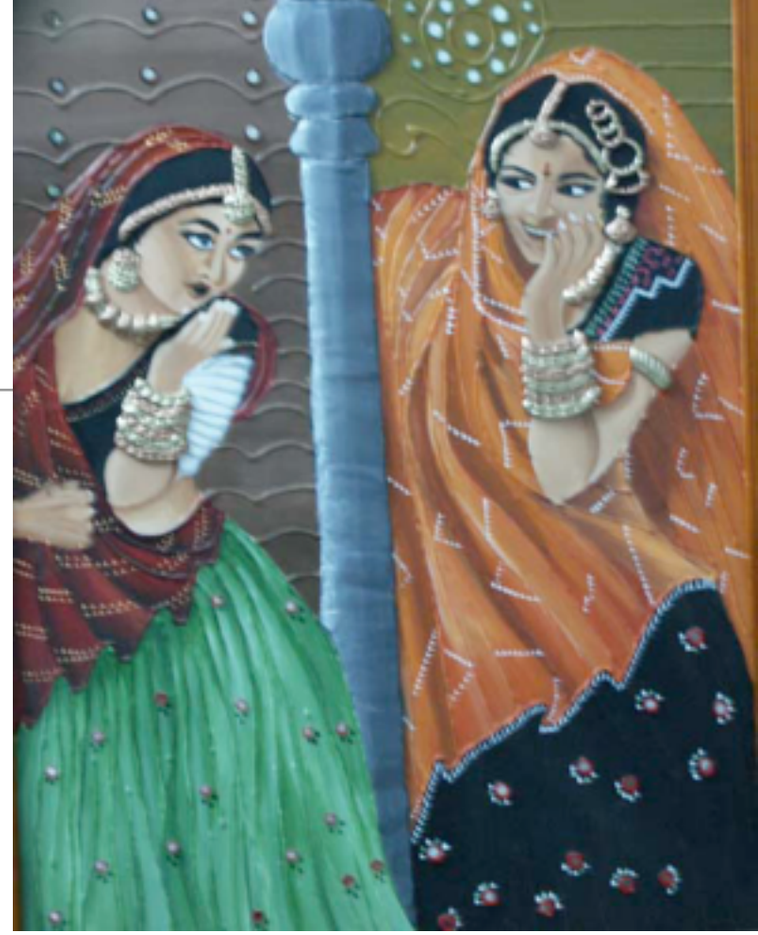


Art number 45

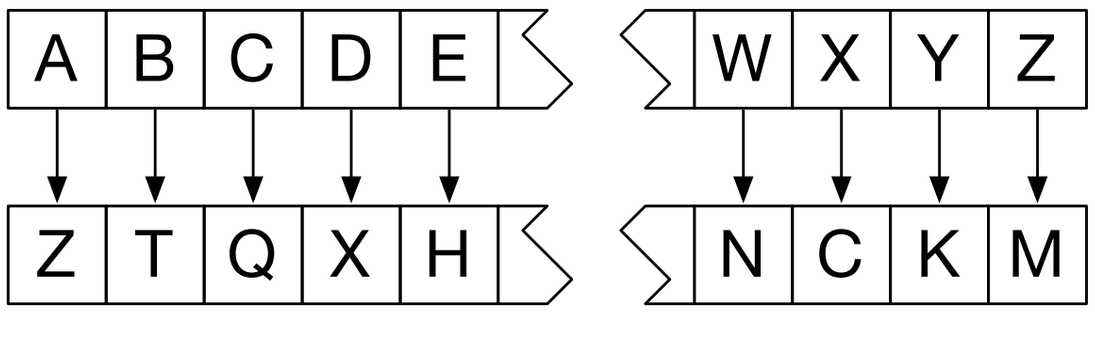
The Kama Sutra recommends that women learn the art of secret writing, to help them conceal their liaisons



Art number 45

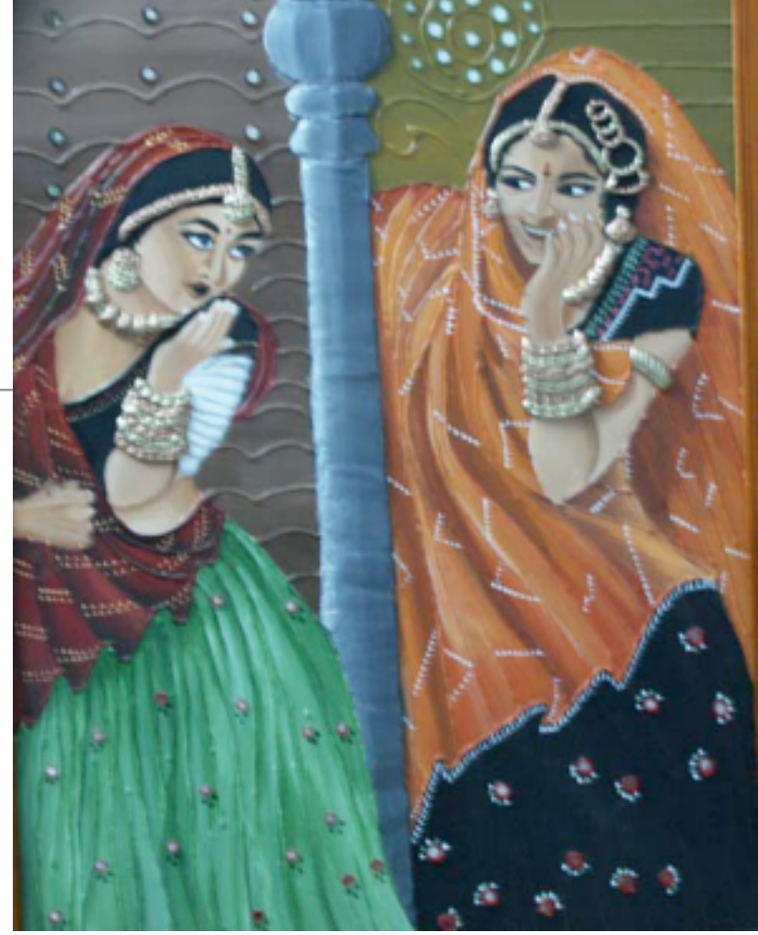


The Kama Sutra recommends that women learn the art of secret writing, to help them conceal their liaisons

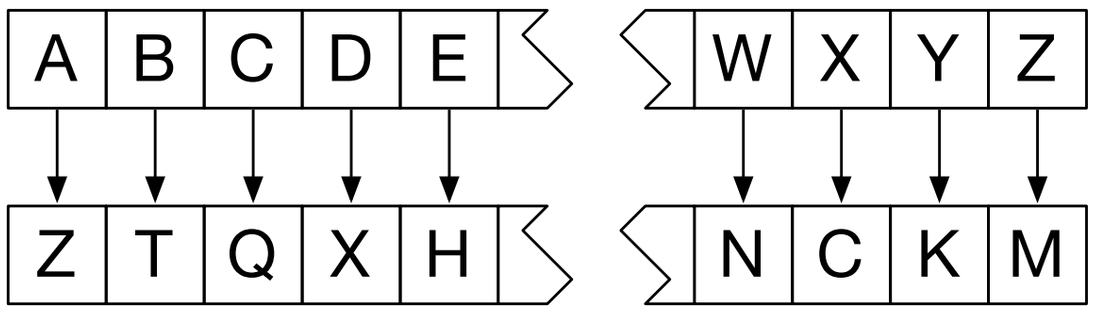


Plaintext: `MEET ME BY THE BANYAN TREE`

Art number 45



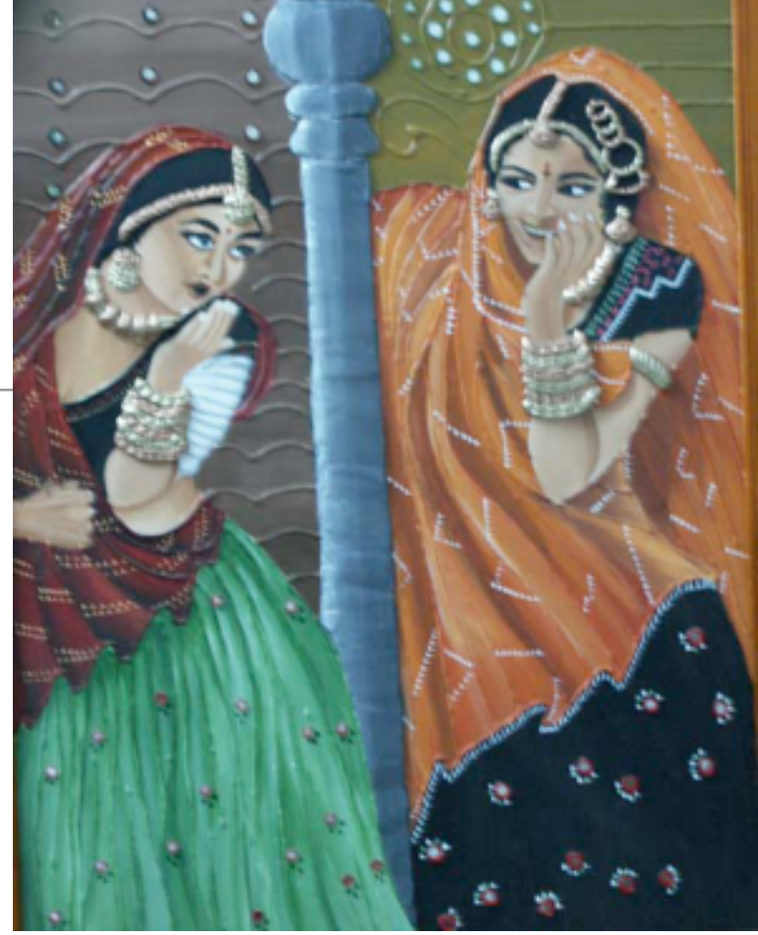
The Kama Sutra recommends that women learn the art of secret writing, to help them conceal their liaisons



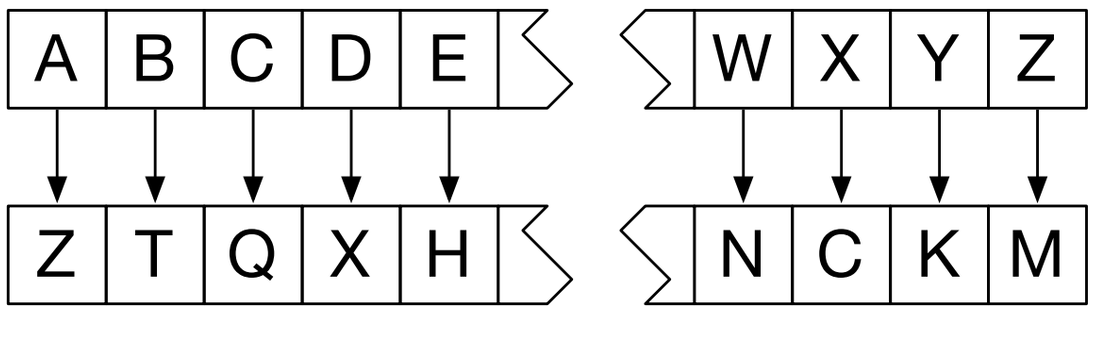
Plaintext: `MEET ME BY THE BANYAN TREE`

Ciphertext: `AHHL AH TK LBH TZSWZS LGHH`

Art number 45



The Kama Sutra recommends that women learn the art of secret writing, to help them conceal their liaisons

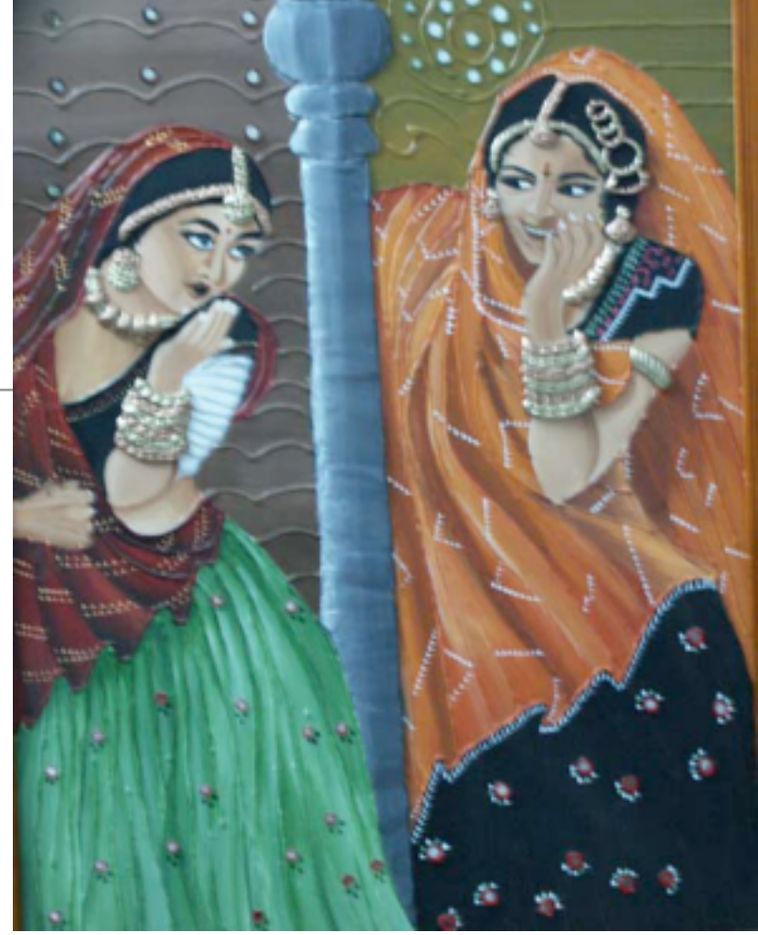


Plaintext: `MEET ME BY THE BANYAN TREE`

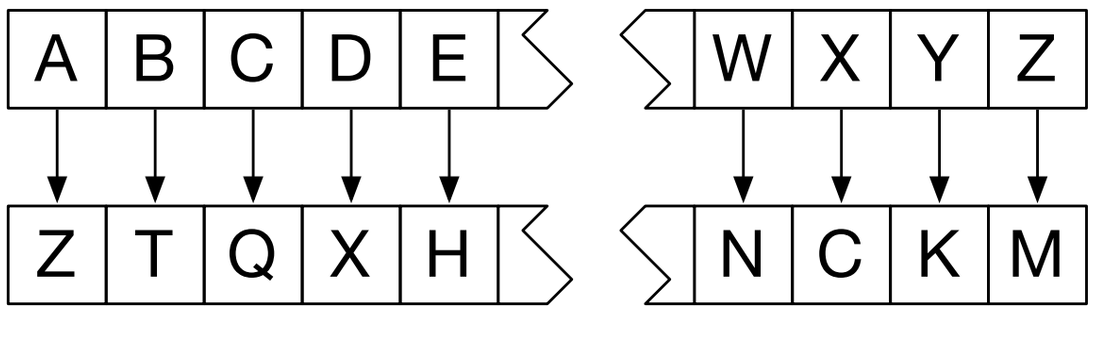
Ciphertext: `AHHL AH TK LBH TZSWZS LGHH`

To decode, look up the letters in the scrambled alphabet

Art number 45



The Kama Sutra recommends that women learn the art of secret writing, to help them conceal their liaisons



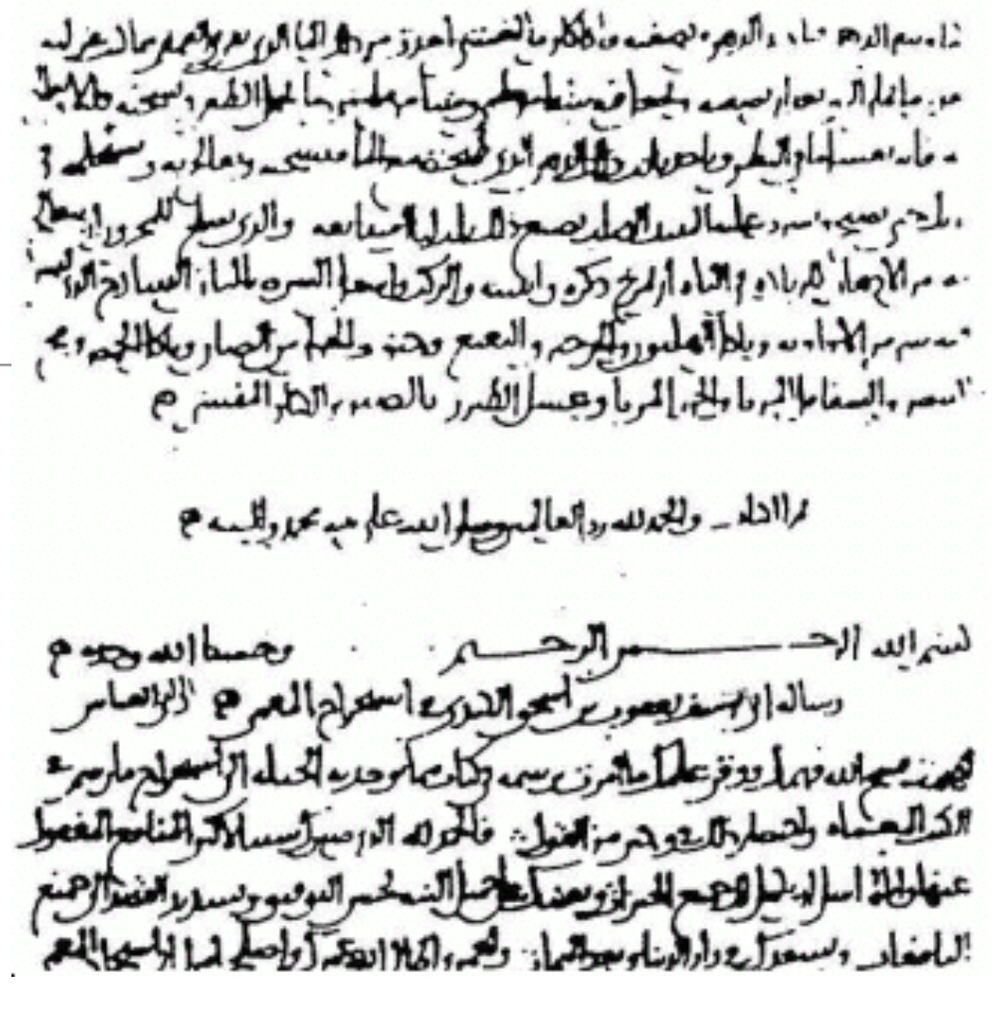
Plaintext: `MEET ME BY THE BANYAN TREE`

Ciphertext: `AHHL AH TK LBH TZSWZS LGHH`

To decode, look up the letters in the scrambled alphabet

26×25×24...3×2×1 = 403,291,461,126,605,635,584,000,000 possibilities

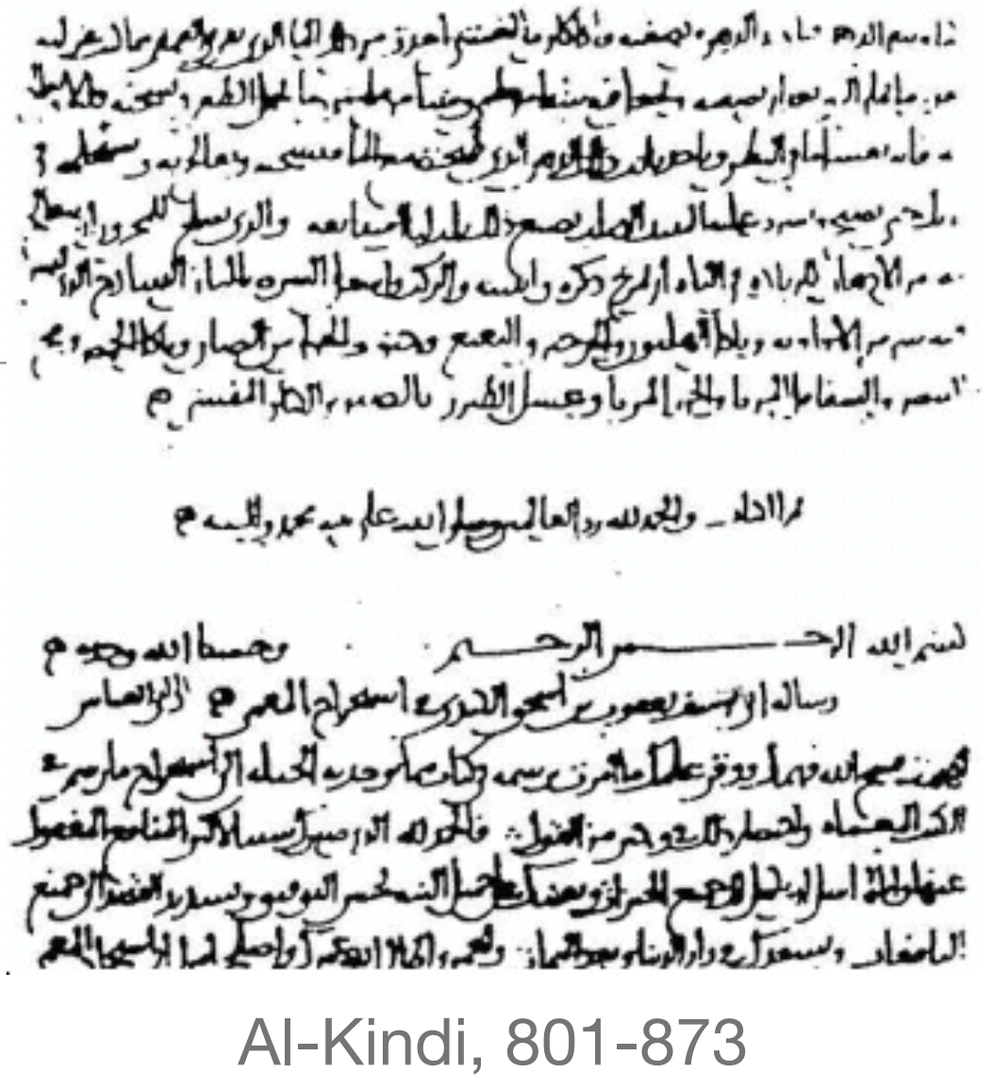
Cracking the code



Al-Kindi, 801-873

Cracking the code

The most common letters are E, T, A, O, I, N

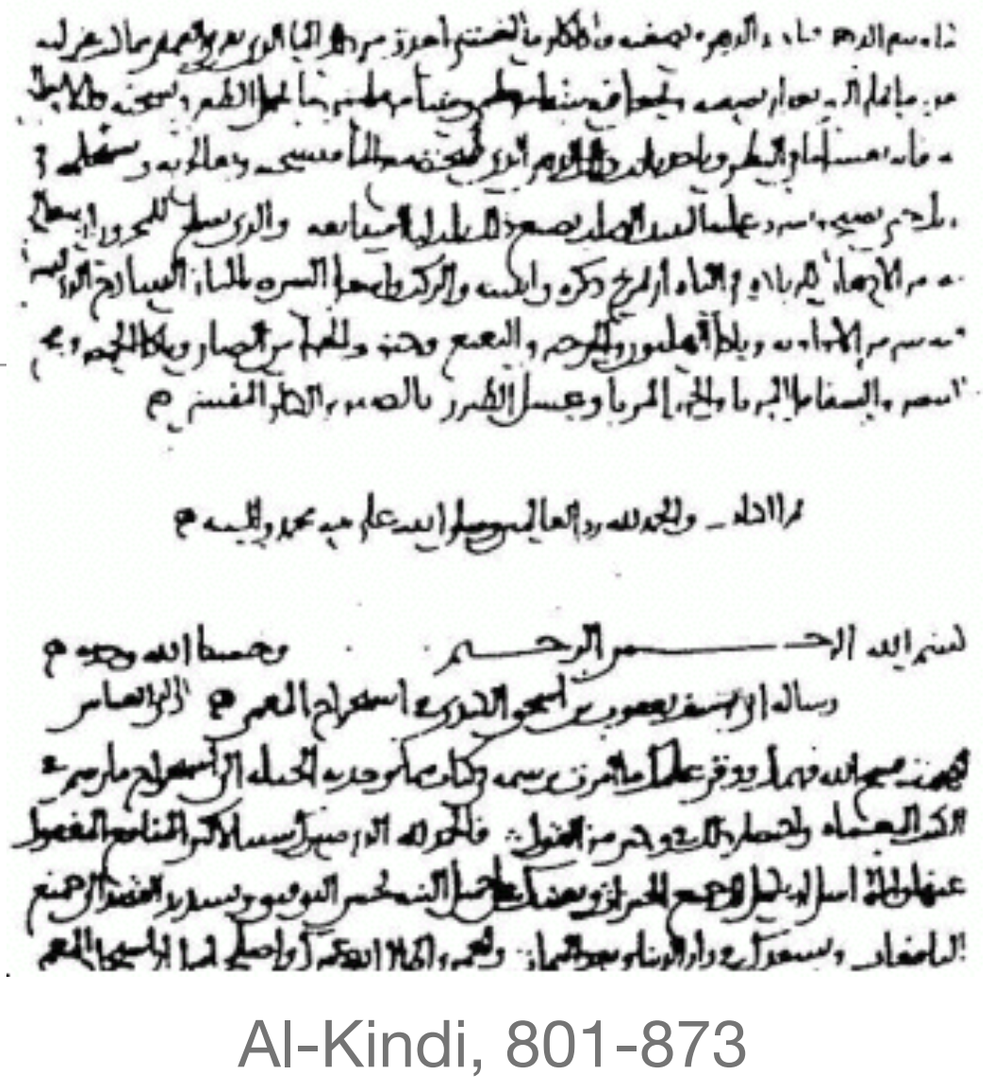


Al-Kindi, 801-873

Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO



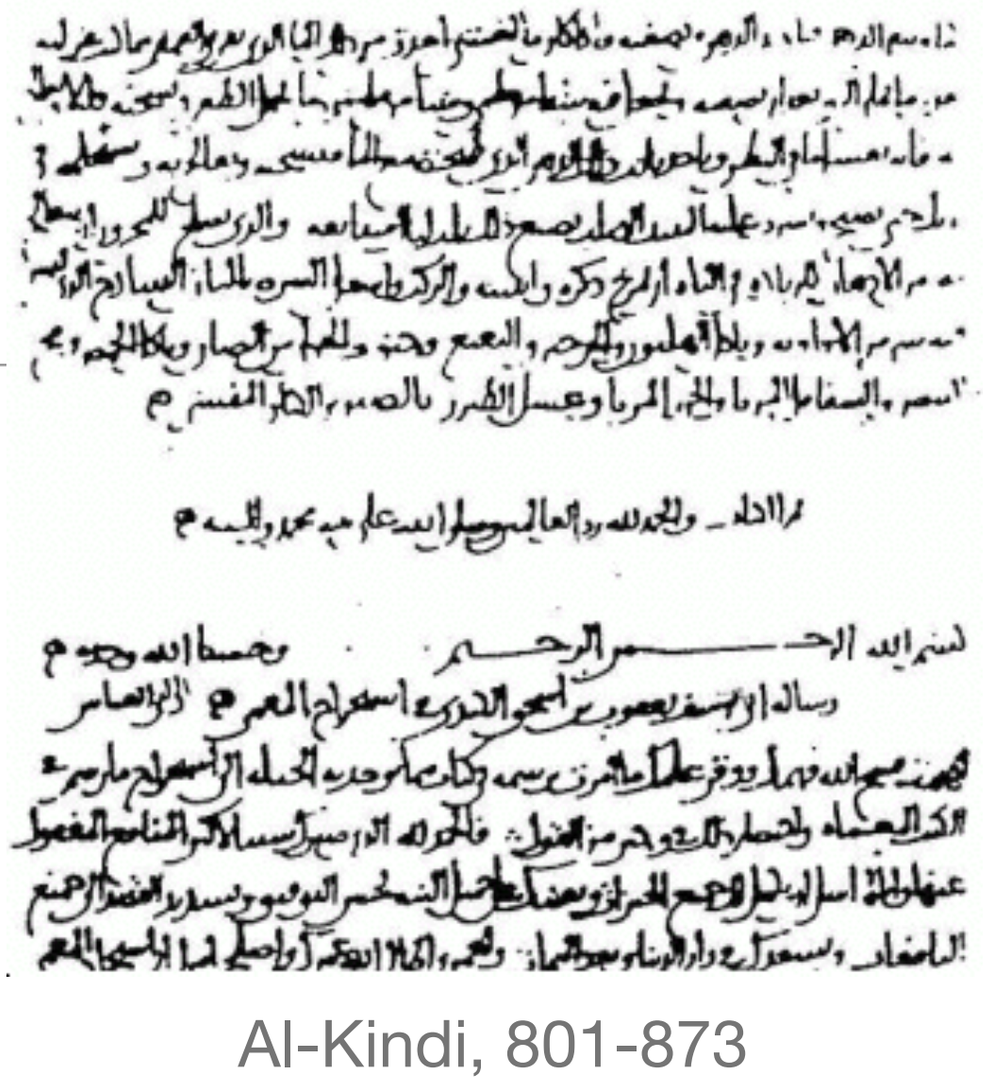
Al-Kindi, 801-873

Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together



Al-Kindi, 801-873

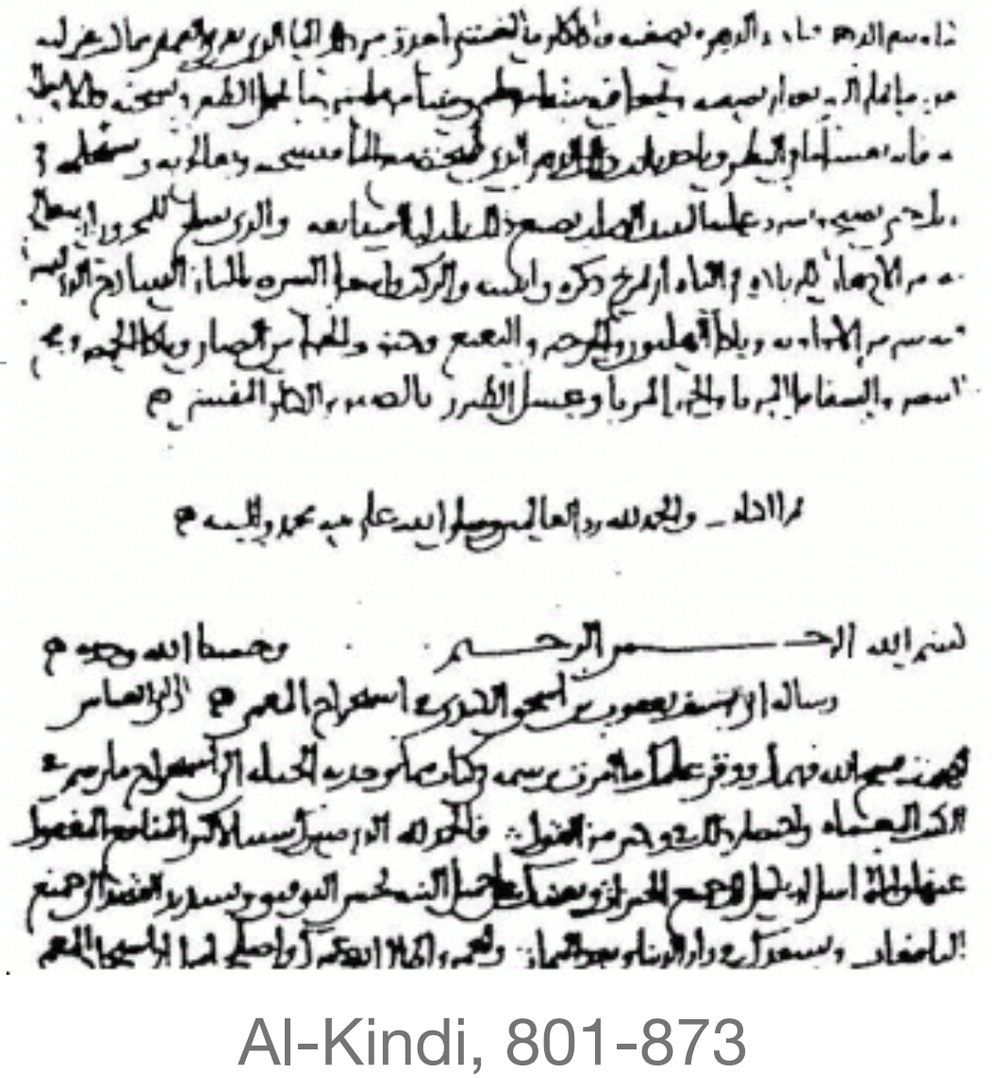
Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot



Al-Kindi, 801-873

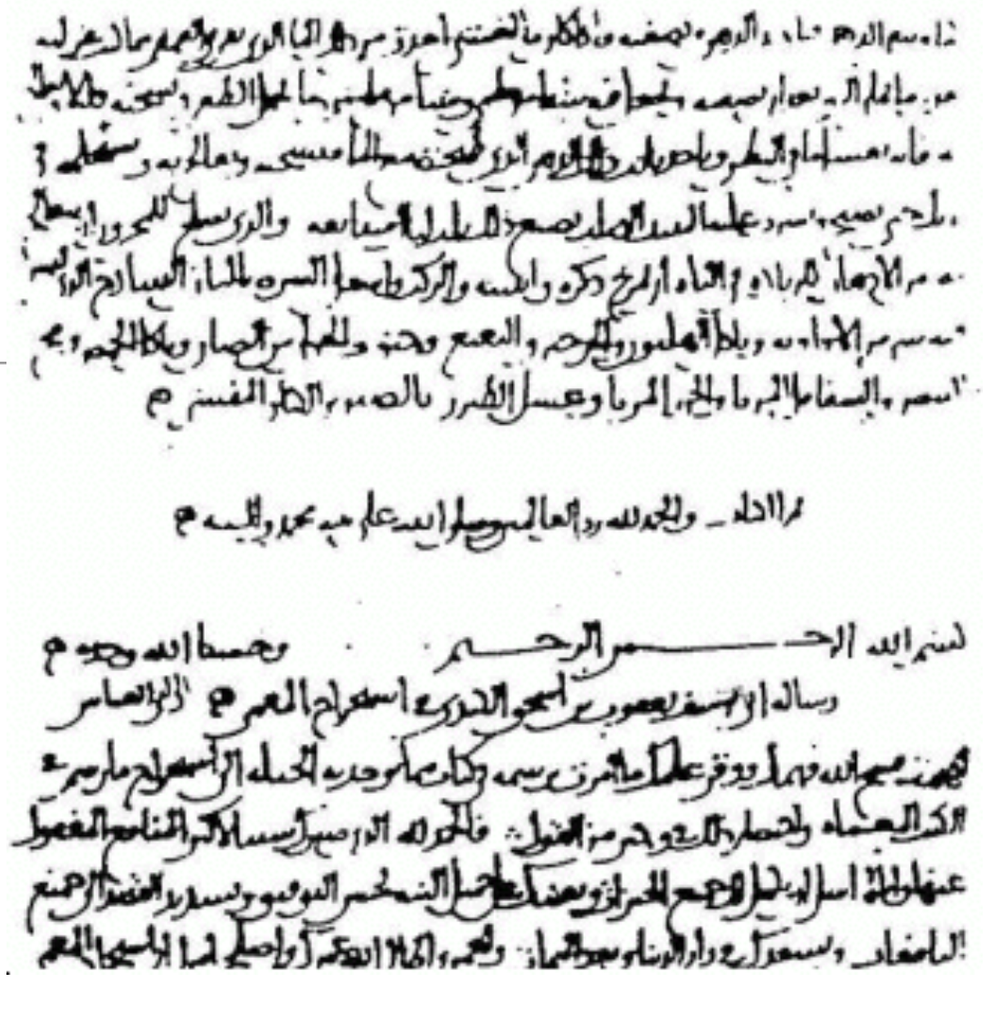
Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot



Al-Kindi, 801-873

```
JNN YBMF NK R HNNW JFQXH QU INXWDEKBG
```

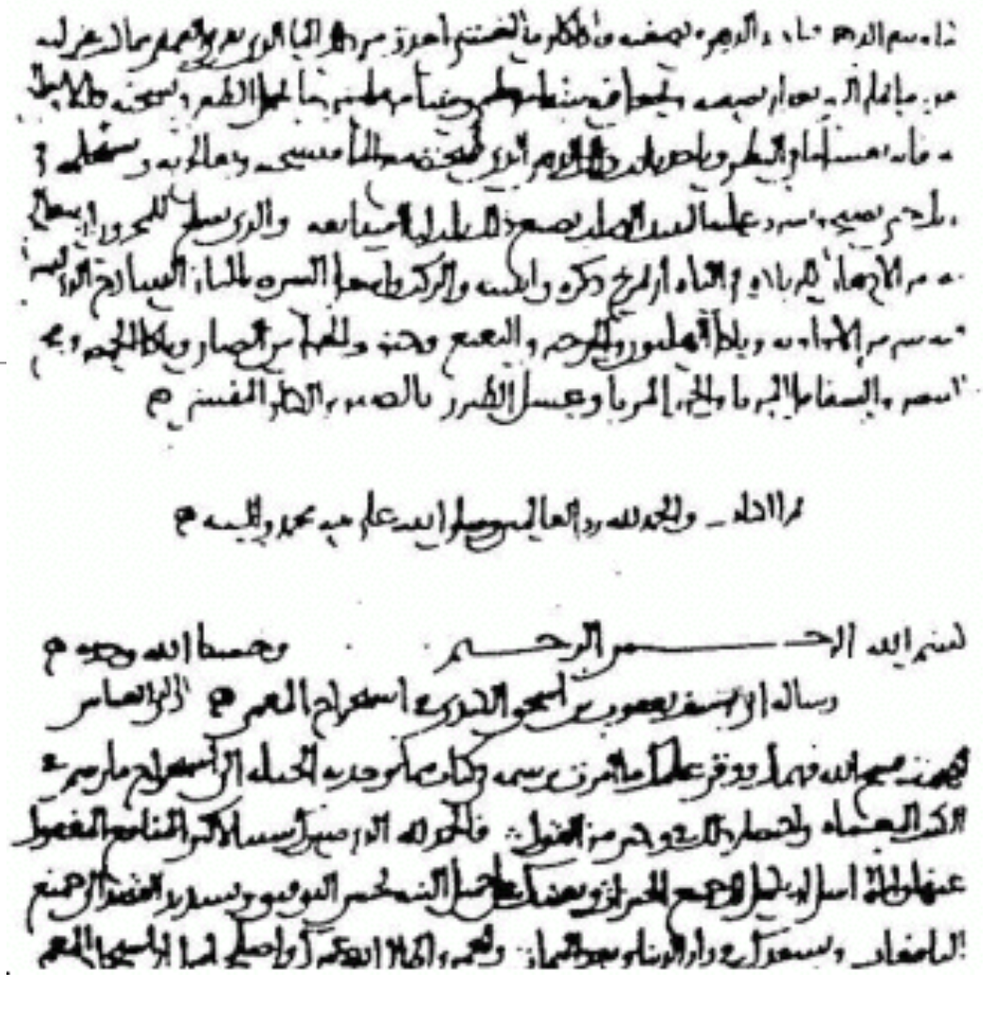
Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot



Al-Kindi, 801-873

```
JNN YBMF NK R HNNW JFQXH QU INXWDEKBG
JOO YBMF OK R HOOW JFQXH QU IOXWDEKBG
```

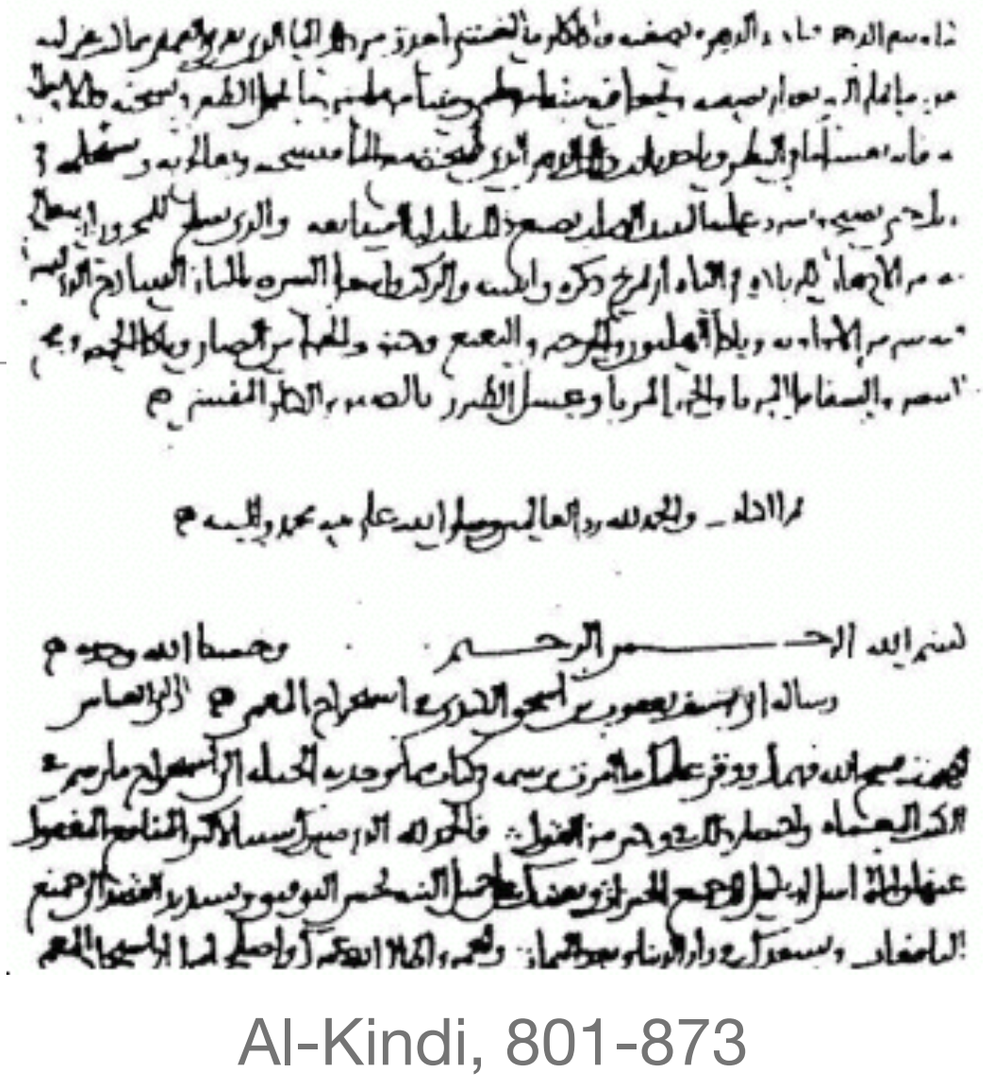
Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot



Al-Kindi, 801-873

```
JNN YBMF NK R HNNW JFQXH QU INXWDEKBG
JOO YBMF OK R HOOW JFQXH QU IOXWDEKBG
TOO YBMF OK R HOOW TFQXH QU IOXWDEKBG
```

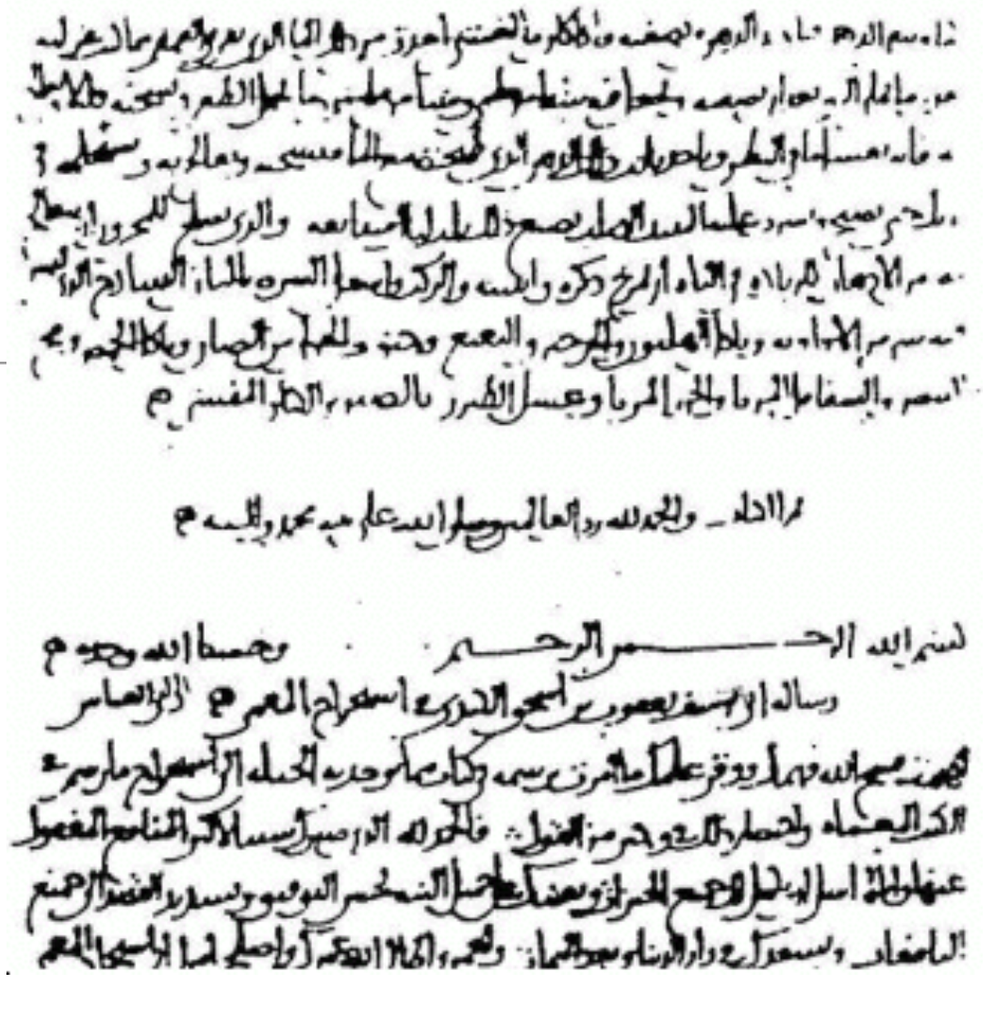
Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot



Al-Kindi, 801-873

```
JNN YBMF NK R HNNW JFQXH QU INXWDEKBG
JOO YBMF OK R HOOW JFQXH QU IOXWDEKBG
TOO YBMF OK R HOOW TFQXH QU IOXWDEKBG
TOO YBMF OF A HOOW TFIXH IS IOXWDEFBG
```

Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot

Al-Kindi, 801-873

```
JNN YBMF NK R HNNW JFQXH QU INXWDEKBG
JOO YBMF OK R HOOW JFQXH QU IOXWDEKBG
TOO YBMF OK R HOOW TFQXH QU IOXWDEKBG
TOO YBMF OF A HOOW TFIXH IS IOXWDEFBG
TOO YBMH OF A GOOD THING IS IONDDEFBG
```

Cracking the code

The most common letters are E, T, A, O, I, N

Doubled letters: LL, EE, SS, OO

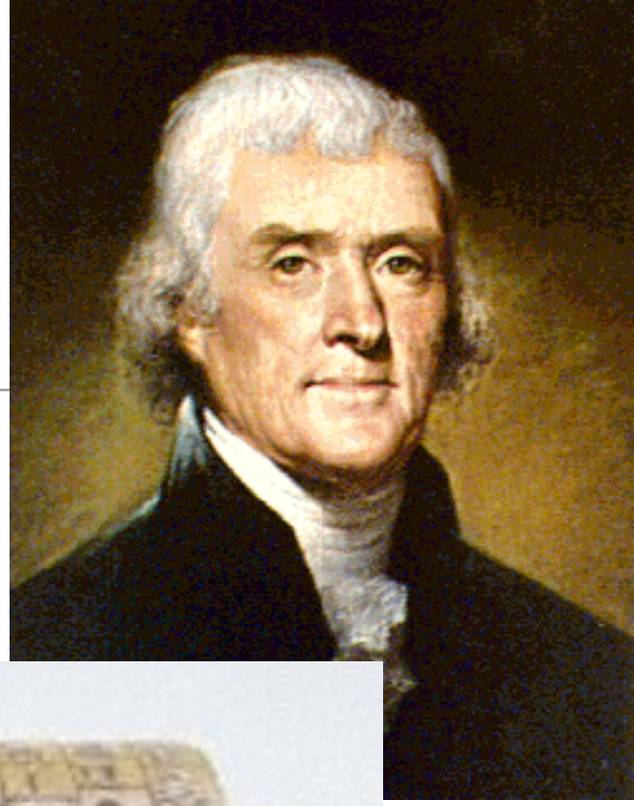
Digrams: letter pairs like QU that go together

Punctuation and spaces help a lot

Al-Kindi, 801-873

JNN YBMF NK R HNNW JFQXH QU INXWDEKBG
JOO YBMF OK R HOOW JFQXH QU IOXWDEKBG
TOO YBMF OK R HOOW TFQXH QU IOXWDEKBG
TOO YBMF OF A HOOW TFIXH IS IOXWDEFBG
TOO YBMH OF A GOOD THING IS IONDDEFBG
TOO MUCH OF A GOOD THING IS WONDERFUL

Jefferson’s cylinder



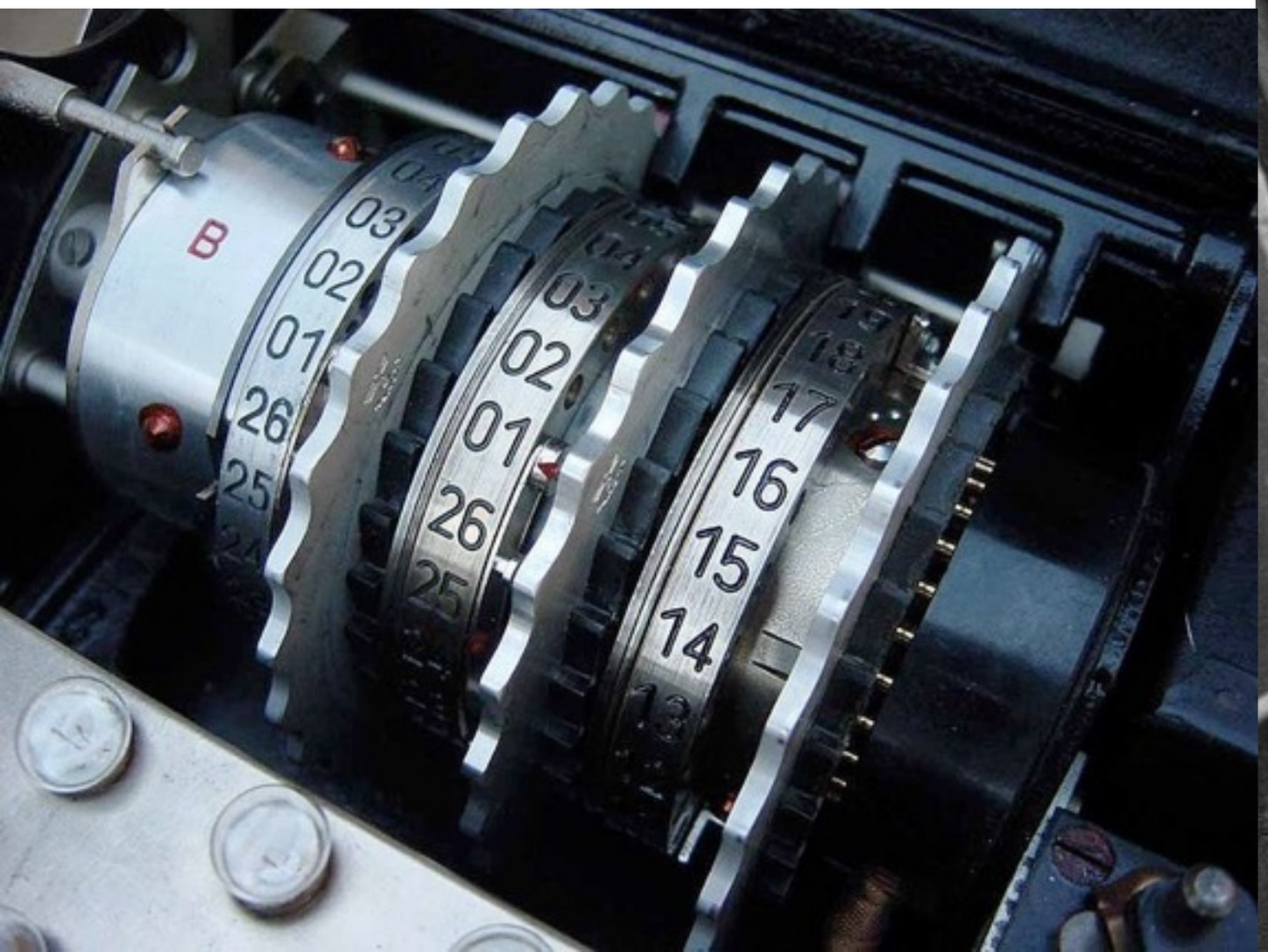
Each disk does a different scramble: the secret key is the order of the disks

The enigma machine

Each letter changes the cipher, affecting the rest of the code, in a different way

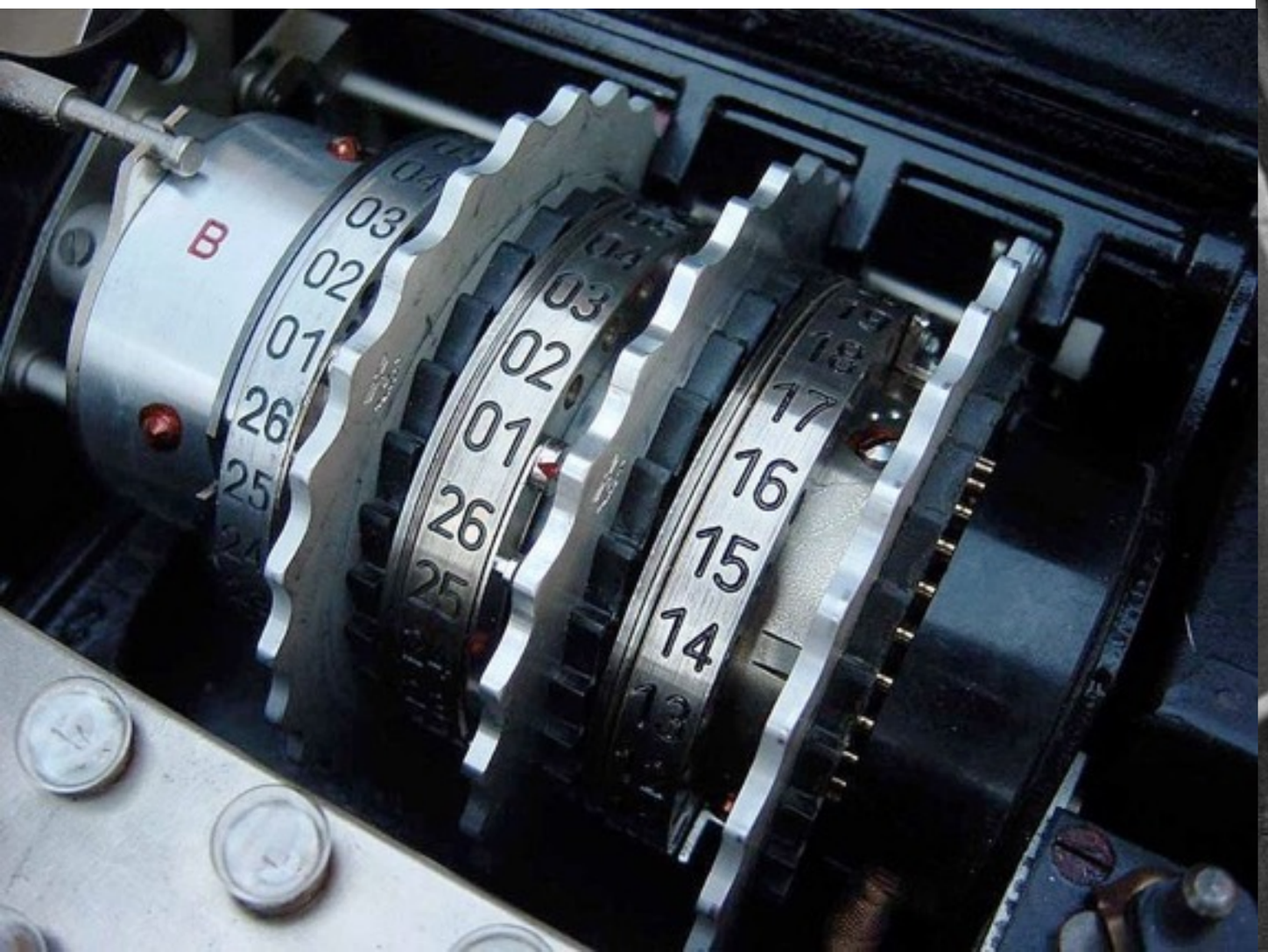
The enigma machine

Each letter changes the cipher, affecting the rest of the code, in a different way



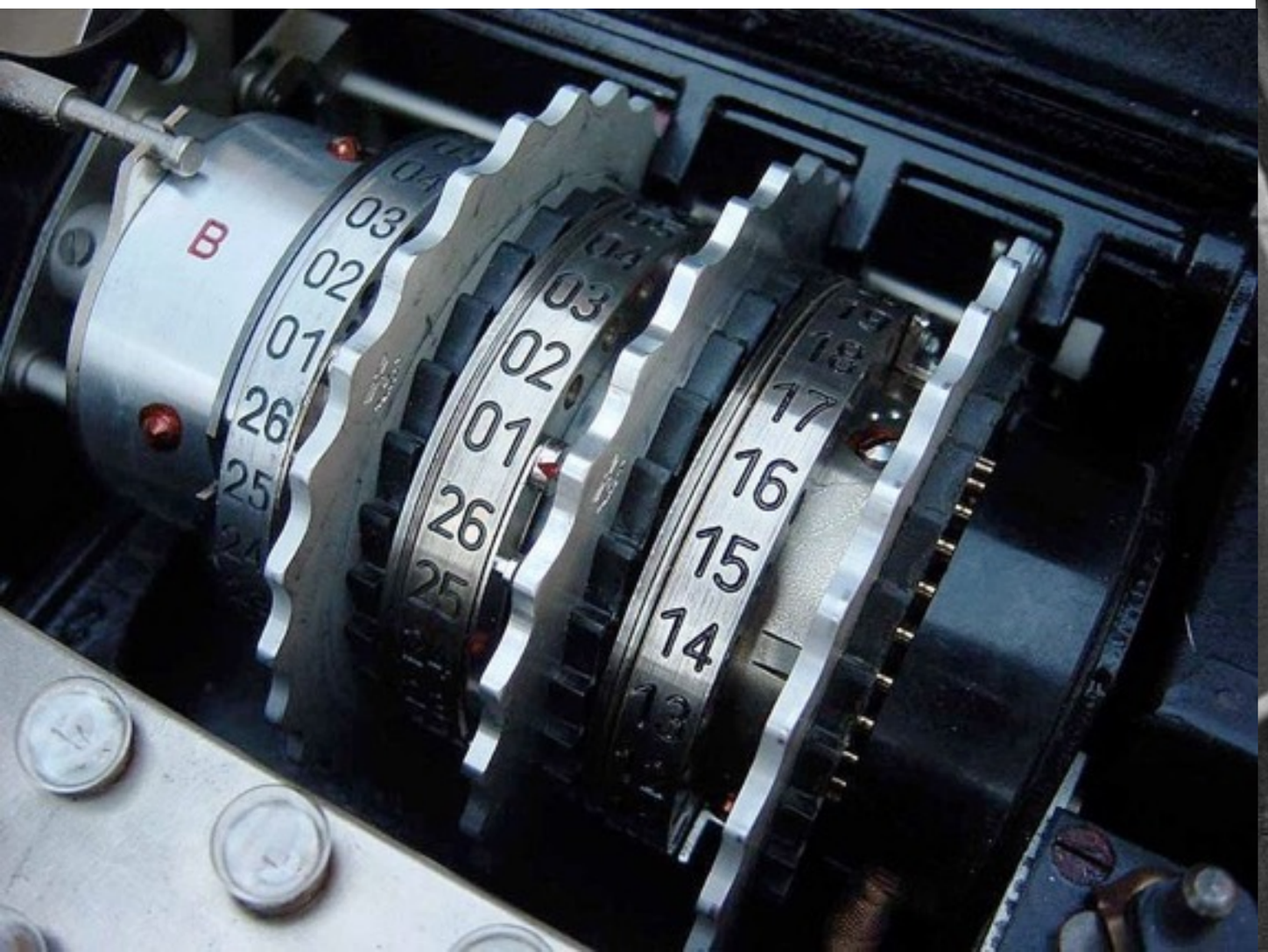
The enigma machine

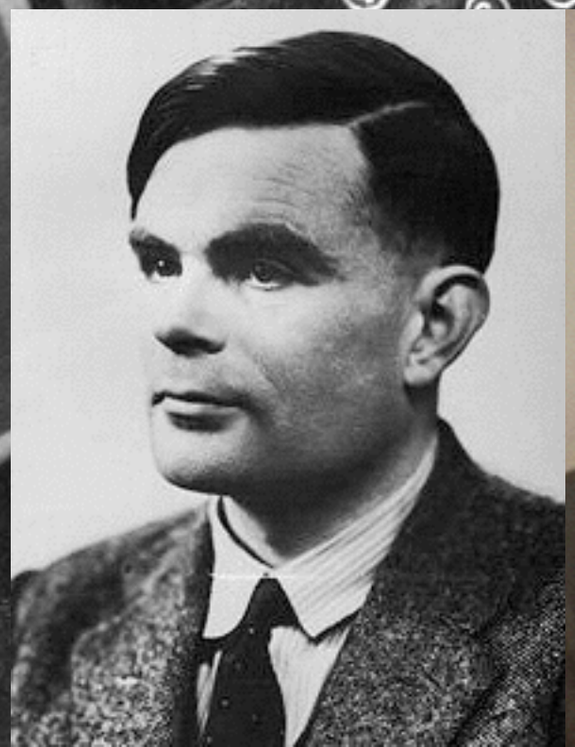
Each letter changes the cipher, affecting the rest of the code, in a different way

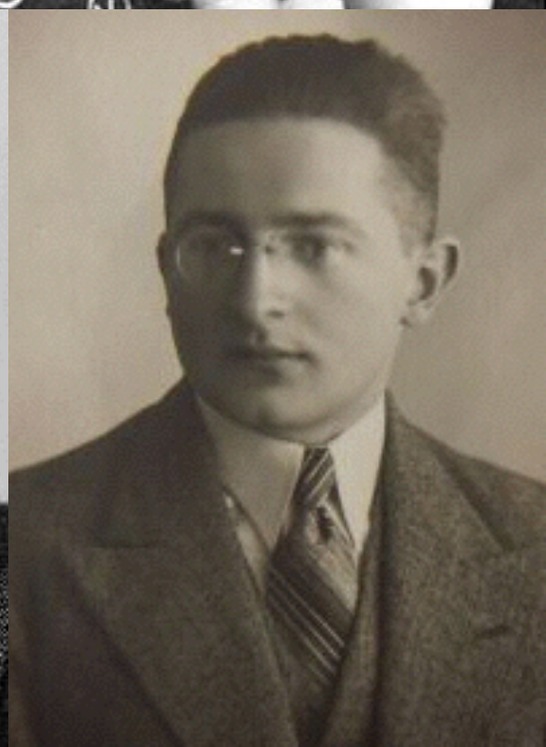


The enigma machine

Each letter changes the cipher, affecting the rest of the code, in a different way

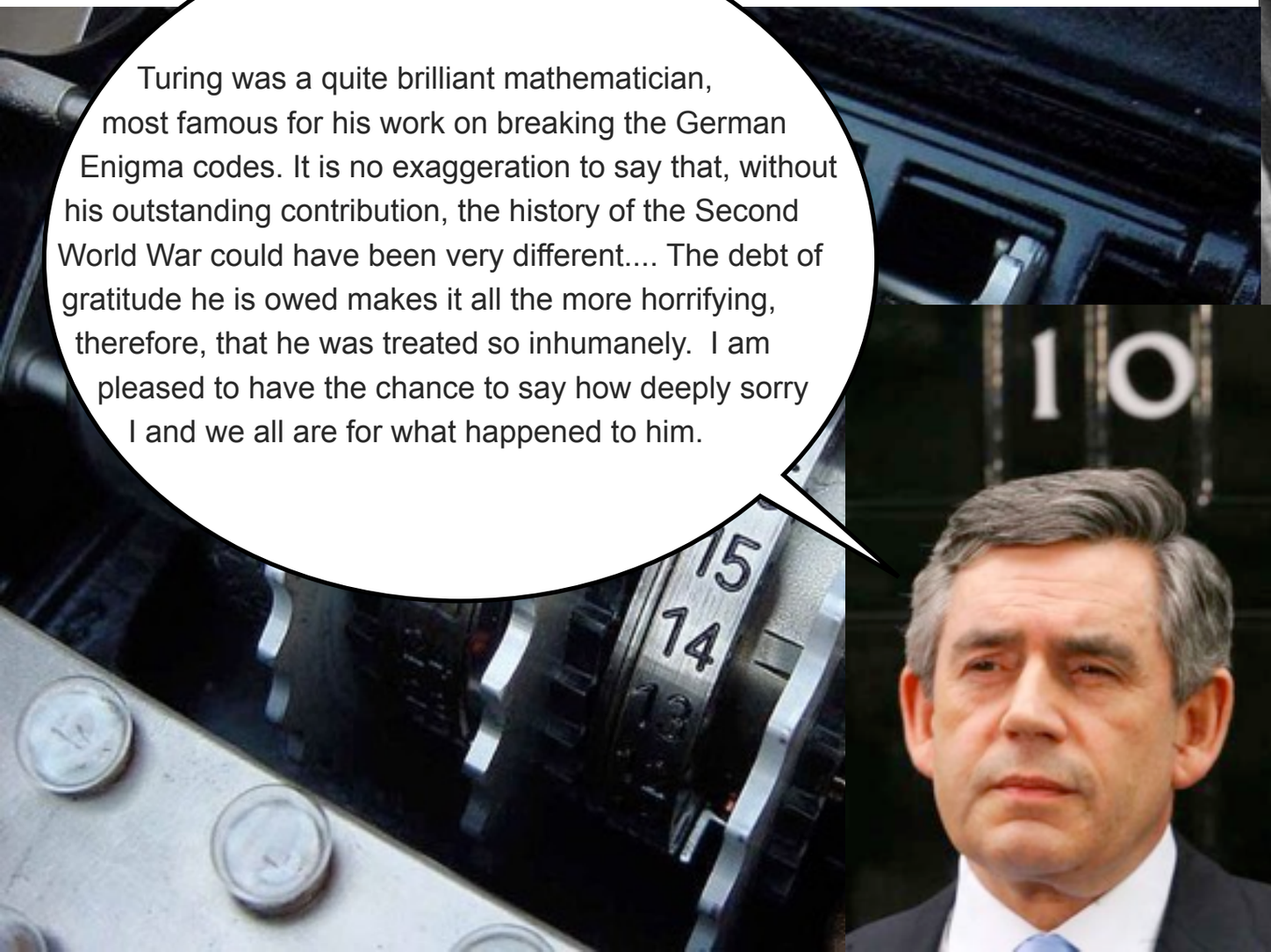




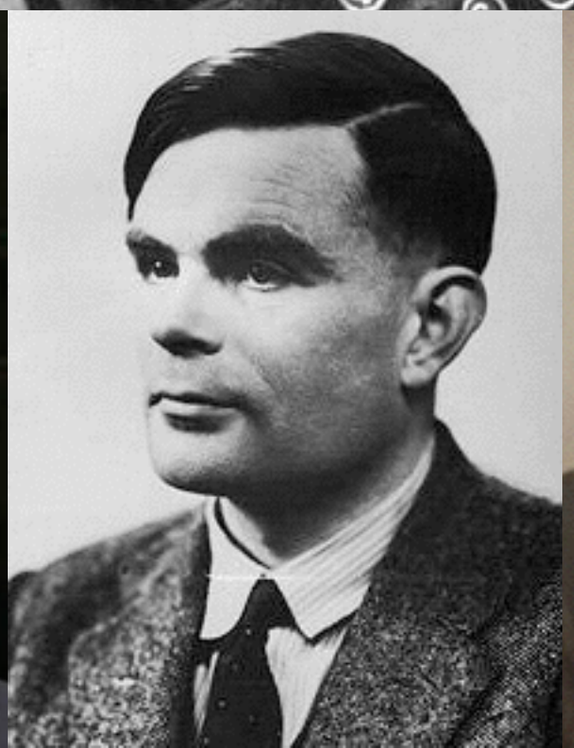


The enigma machine

Each letter changes the cipher, affecting the rest of the code, in a different way



Turing was a quite brilliant mathematician, most famous for his work on breaking the German Enigma codes. It is no exaggeration to say that, without his outstanding contribution, the history of the Second World War could have been very different.... The debt of gratitude he is owed makes it all the more horrifying, therefore, that he was treated so inhumanely. I am pleased to have the chance to say how deeply sorry I and we all are for what happened to him.



True security—but at a cost

True security—but at a cost

One-time pad:

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3
pad:	1	4	1	5	9	2	6	5

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3
pad:	1	4	1	5	9	2	6	5
ciphertext:	2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

True security—but at a cost

One-time pad:

ciphertext:

2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

True security—but at a cost

One-time pad:

pad:	1	4	1	5	9	2	6	5
ciphertext:	2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3
pad:	1	4	1	5	9	2	6	5
ciphertext:	2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3
pad:	1	4	1	5	9	2	6	5
ciphertext:	2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

If the pad is truly random, the ciphertext is random too:
No pattern that the codebreaker can discover

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3
pad:	1	4	1	5	9	2	6	5
ciphertext:	2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

If the pad is truly random, the ciphertext is random too:
No pattern that the codebreaker can discover

But never use the same pad twice!

True security—but at a cost

One-time pad:

plaintext:	1	0	0	6	4	1	0	3
pad:	1	4	1	5	9	2	6	5
ciphertext:	2	4	1	1	3	3	6	8

note: add mod 10 (no carries!)

If the pad is truly random, the ciphertext is random too:
No pattern that the codebreaker can discover

But never use the same pad twice!

Must share one digit of pad for every digit of message,
and must do this in a physically secure way.

A more tangled web

A more tangled web

All of these methods encrypt one letter at a time

A more tangled web

All of these methods encrypt one letter at a time

This lets us break them one letter at a time... like passwords in the movies

A more tangled web

All of these methods encrypt one letter at a time

This lets us break them one letter at a time... like passwords in the movies

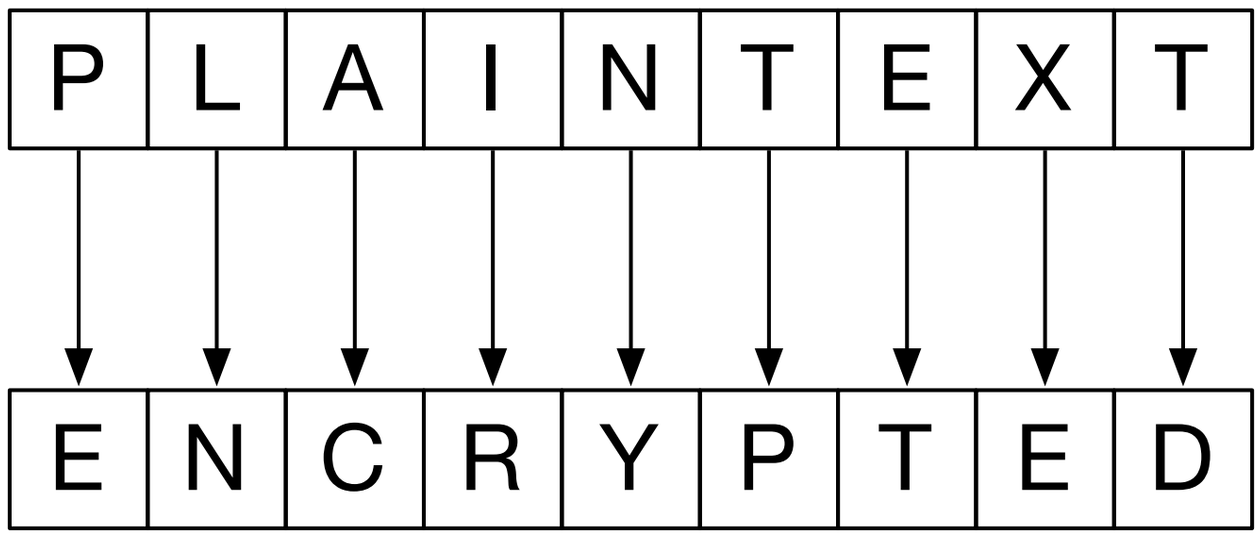
What we need is a method where every letter affects every other letter:

A more tangled web

All of these methods encrypt one letter at a time

This lets us break them one letter at a time... like passwords in the movies

What we need is a method where every letter affects every other letter:

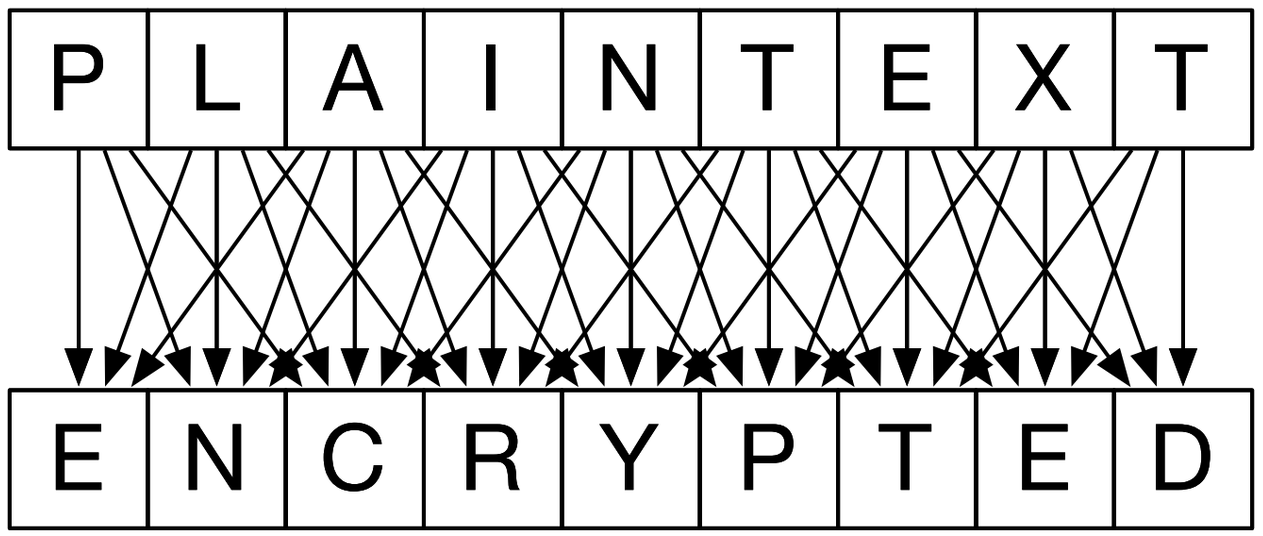


A more tangled web

All of these methods encrypt one letter at a time

This lets us break them one letter at a time... like passwords in the movies

What we need is a method where every letter affects every other letter:



Cyclic arithmetic

Cyclic arithmetic

"*x* mod *N*" is the remainder when we divide *x* by *N*:
for instance, *x* mod 10 is *x*'s 1s digit

Cyclic arithmetic

"*x* mod *N*" is the remainder when we divide *x* by *N*:
for instance, *x* mod 10 is *x*'s 1s digit

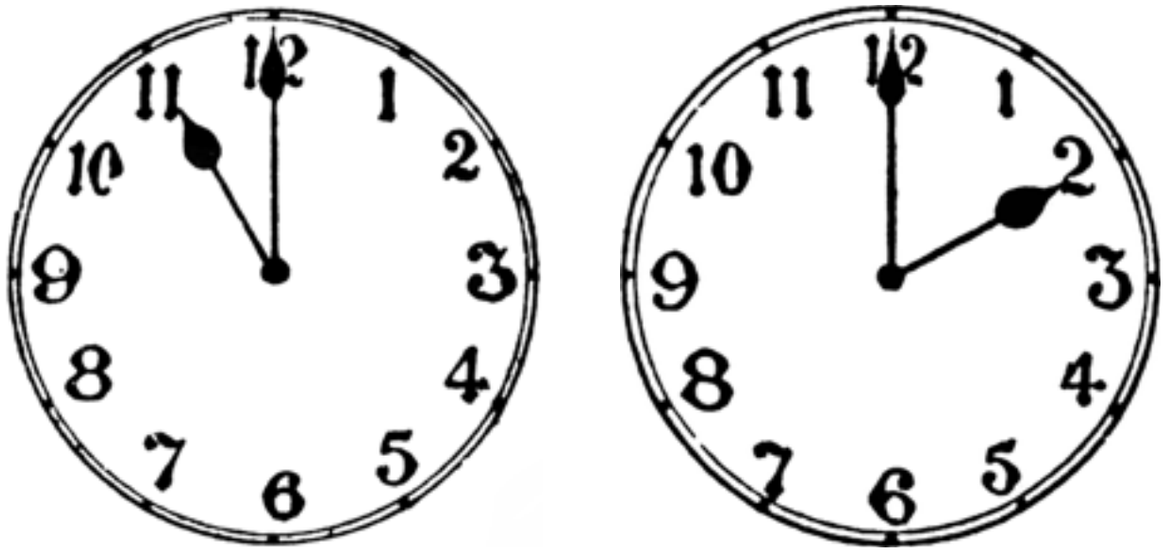
6+5 mod 10 = 1

Cyclic arithmetic

"*x* mod *N*" is the remainder when we divide *x* by *N*:
for instance, *x* mod 10 is *x*'s 1s digit

6+5 mod 10 = 1

11+3 mod 12 = 2



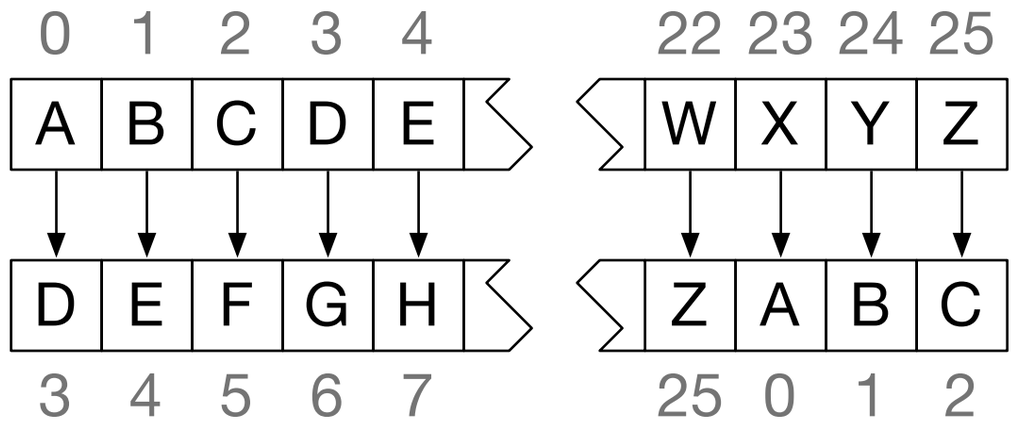
Cyclic arithmetic

"*x* mod *N*" is the remainder when we divide *x* by *N*:
for instance, *x* mod 10 is *x*'s 1s digit

6+5 mod 10 = 1

11+3 mod 12 = 2

24+3 mod 26 = 1: Y⇒B



Cyclic arithmetic

"*x* mod *N*" is the remainder when we divide *x* by *N*:
for instance, *x* mod 10 is *x*'s 1s digit

6+5 mod 10 = 1

11+3 mod 12 = 2

24+3 mod 26 = 1: Y⇒B

multiplication mod 5: 2×3 = 1

×	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Cyclic arithmetic

"*x* mod *N*" is the remainder when we divide *x* by *N*:
for instance, *x* mod 10 is *x*'s 1s digit

6+5 mod 10 = 1

11+3 mod 12 = 2

24+3 mod 26 = 1: Y⇒B

multiplication mod 5: 2×3 = 1

75,764 mod 1,000 = 764

RSA public-key encryption

RSA public-key encryption

RSA public-key encryption

Anyone can send a message to Alice,
but only she can decrypt it

RSA public-key encryption

Anyone can send a message to Alice,
but only she can decrypt it

Alice publishes her public key: *e*, *N*

RSA public-key encryption

Anyone can send a message to Alice,
but only she can decrypt it

Alice publishes her public key: *e*, *N*

To send Alice a message *m*,
Bob encrypts it as m^e mod *N*

RSA public-key encryption

Anyone can send a message to Alice,
but only she can decrypt it

Alice publishes her public key: *e*, *N*

To send Alice a message *m*,
Bob encrypts it as m^e mod *N*

For example: *e*=3 and *N*=1,000

RSA public-key encryption

Anyone can send a message to Alice,
but only she can decrypt it

Alice publishes her public key: e, N

To send Alice a message m,
Bob encrypts it as $m^e \bmod N$

For example: e=3 and N=1,000

To send Alice m=**413**, Bob encrypts it as

$$413^3 = 413 \times 413 \times 413 = 70{,}444{,}\mathbf{997}$$

RSA public-key encryption

RSA public-key encryption

Alice's public key: e=3, N=1,000

To send Alice m=**413**, Bob encrypts it as

$$\mathbf{413}^3 = 413\times413\times413 = 70{,}444{,}\mathbf{997}$$

RSA public-key encryption

Alice's public key: e=3, N=1,000

To send Alice m=**413**, Bob encrypts it as

$$\mathbf{413}^3 = 413\times413\times413 = 70{,}444{,}\mathbf{997}$$

Only Alice knows that if we raise the encrypted message to the **67**th power, she gets the original message m back!

RSA public-key encryption

Alice's public key: e=3, N=1,000

To send Alice m=**413**, Bob encrypts it as

$$\mathbf{413}^3 = 413\times413\times413 = 70{,}444{,}\mathbf{997}$$

Only Alice knows that if we raise the encrypted message to the **67**th power, she gets the original message m back!

$$\mathbf{997}^{67} =$$
$$81766537396264378675981369924764615863078578752890456861822811772191993279453137376140622894950430471674461990282376875754836451383036420378360358129864380547637845421151481946569591385634343038032\mathbf{413}$$

RSA public-key encryption

Alice's public key: e=3, N=1,000

To send Alice m=**413**, Bob encrypts it as

$$\mathbf{413}^3 = 413\times413\times413 = 70{,}444{,}\mathbf{997}$$

Only Alice knows that if we raise the encrypted message to the **67**th power, she gets the original message m back!

caveat: m must be odd and not a multiple of 5

$$\mathbf{997}^{67} =$$
817665373962643786759813699247646158630785787528904568618228
117721919932794531373761406228949504304716744619902823768572
754836451383036420378360358129864380547637845421151481946569
5913856343430380**32413**

RSA public-key encryption

RSA public-key encryption

Alice's public key: *e*=3, *N*=1000

RSA public-key encryption

Alice's public key: e=3, N=1000

Her private key: d=67

RSA public-key encryption

Alice's public key: e=3, N=1000

Her private key: d=67

Can we crack her code?

RSA public-key encryption

Alice's public key: e=3, N=1000

Her private key: d=67

Can we crack her code?

Can we get d from e and N?

RSA public-key encryption

Alice's public key: e=3, N=1000

Her private key: d=67

Can we crack her code?

Can we get d from e and N?

We can do this if we can factor N, break it down into a product of prime numbers: 2, 3, 5, 7, 11, 13, 17, 19...

RSA public-key encryption

Alice's public key: e=3, N=1000

Her private key: d=67

Can we crack her code?

Can we get d from e and N?

We can do this if we can factor N, break it down into a product of prime numbers: 2, 3, 5, 7, 11, 13, 17, 19...

For instance, 1000 = 2×2×2×5×5×5

Easy in one direction...

Easy in one direction...

Multiplying is easy

Easy in one direction...

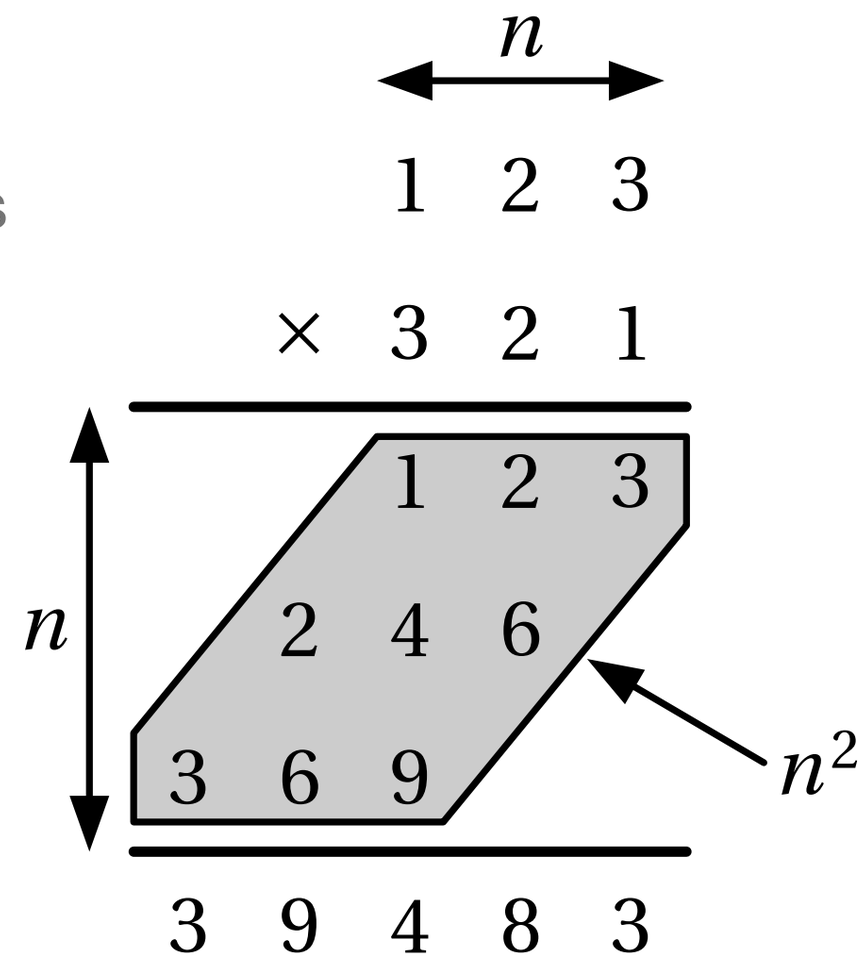
Multiplying is easy

We have an *algorithm* that multiplies n-digit numbers
in about $n \times n = n^2$ time

Easy in one direction...

Multiplying is easy

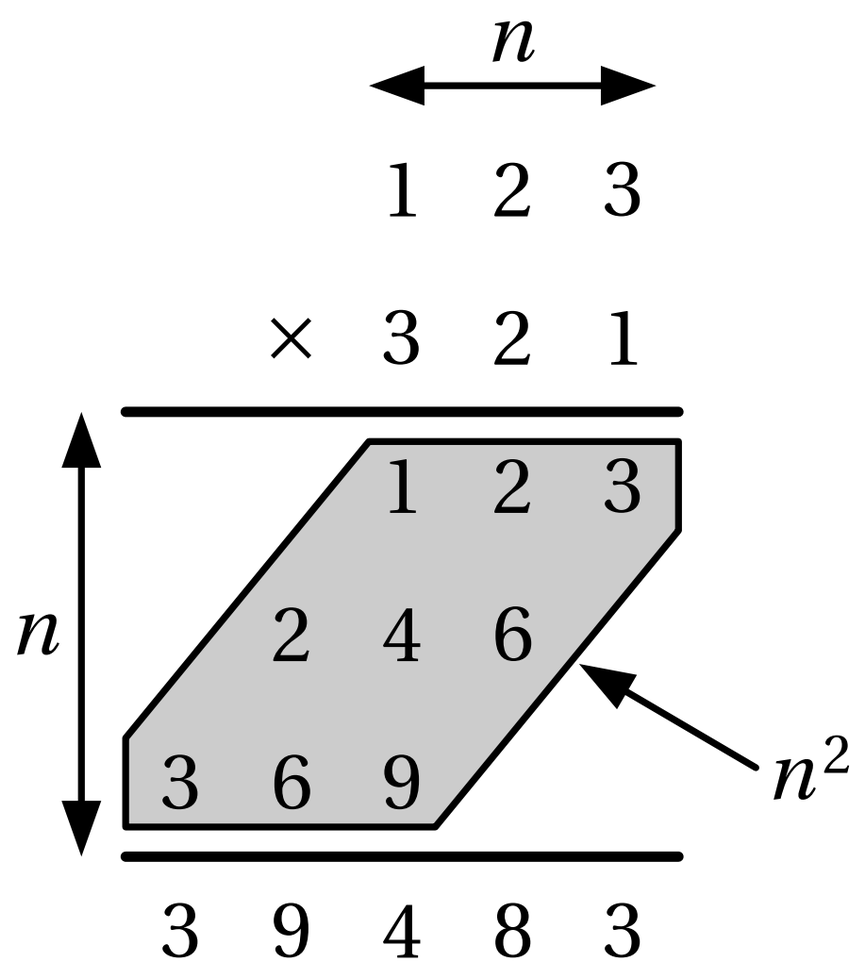
We have an *algorithm* that multiplies n-digit numbers in about $n \times n = n^2$ time



Easy in one direction...

Multiplying is easy

We have an *algorithm* that multiplies n-digit numbers in about $n \times n = n^2$ time



Multiplication is in the class P of problems that can be solved in *polynomial time*

...but hard in the other

...but hard in the other

Is $2^{67}-1$ = 147,573,952,588,676,412,927 a prime?

...but hard in the other

Is $2^{67}-1$ = 147,573,952,588,676,412,927 a prime?

No, because it's 193707721×761838257287

...but hard in the other

Is $2^{67}-1$ = 147,573,952,588,676,412,927 a prime?

No, because it's 193707721×761838257287

Factoring is in the class NP of problems where we can *check* a solution in polynomial time...

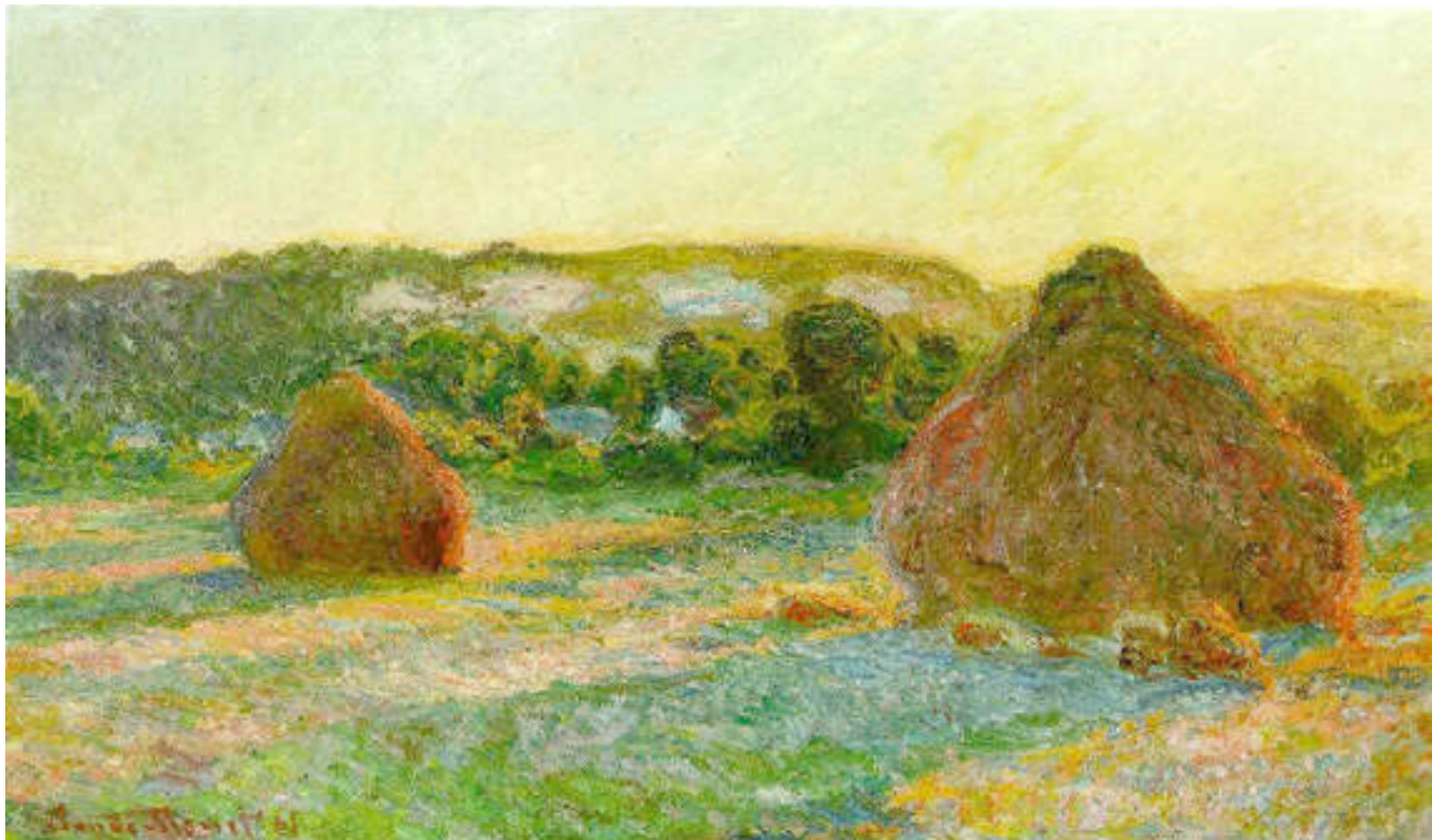
...but hard in the other

Is $2^{67}-1$ = 147,573,952,588,676,412,927 a prime?

No, because it's 193707721×761838257287

Factoring is in the class NP of problems where we can *check* a solution in polynomial time...

...but that doesn't mean we can find one!



...but hard in the other

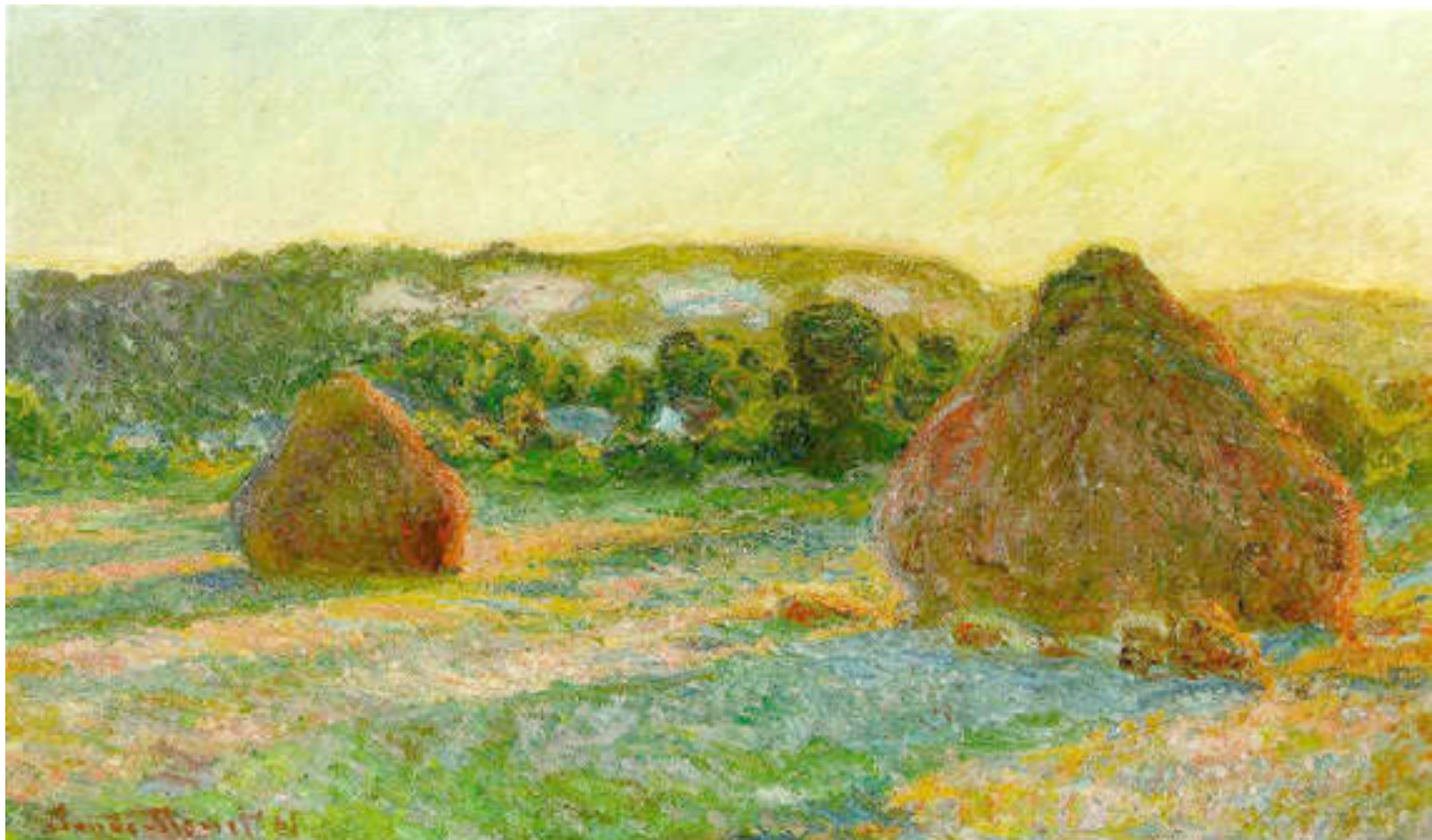
Is $2^{67}-1$ = 147,573,952,588,676,412,927 a prime?

No, because it's 193707721×761838257287

Factoring is in the class NP of problems where we can *check* a solution in polynomial time...

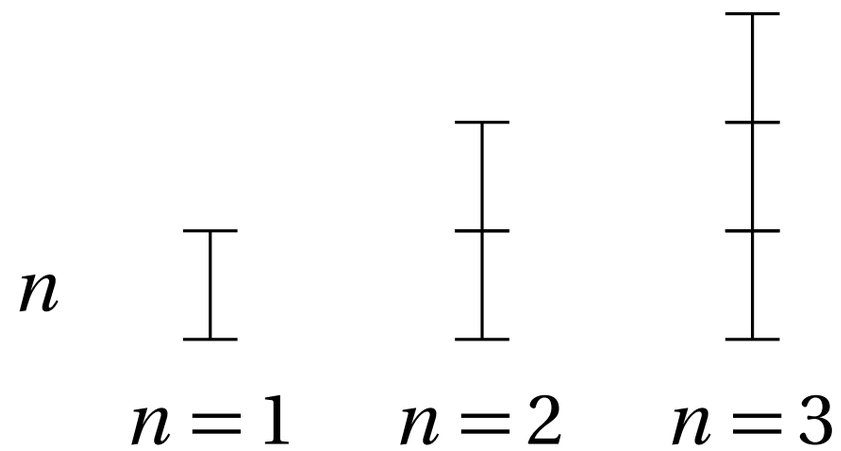
...but that doesn't mean we can find one!

We believe (but don't know) that factoring *n*-digit numbers takes *exponential* time

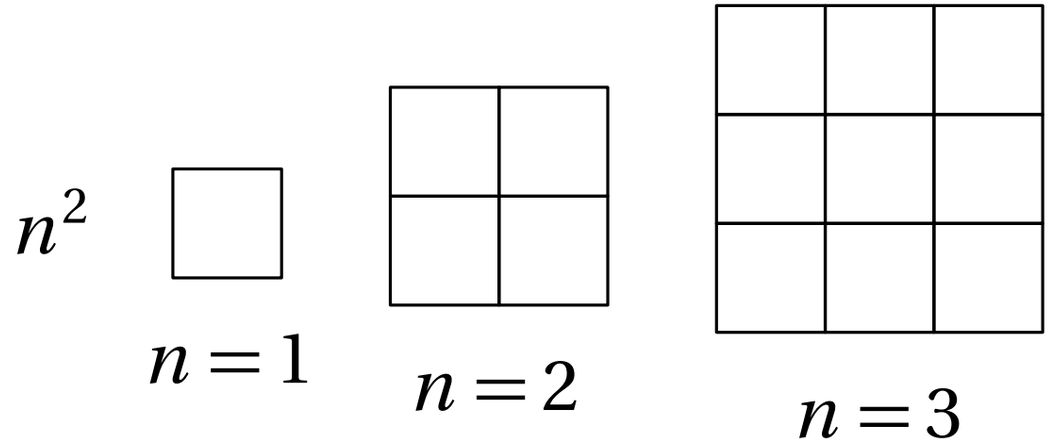


Polynomials don’t grow too badly as n grows...

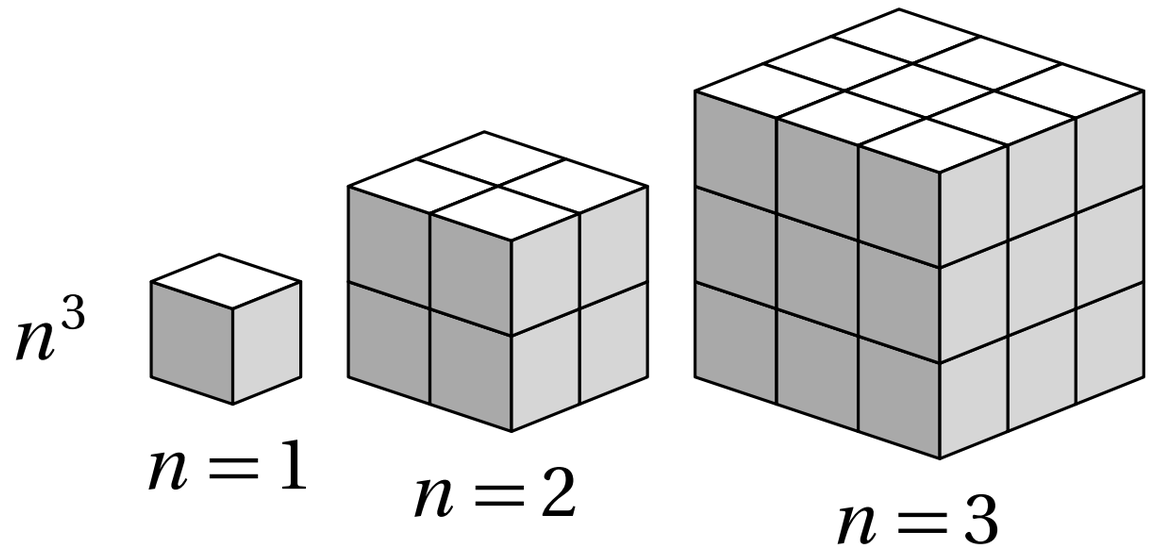
Polynomials don’t grow too badly as n grows...



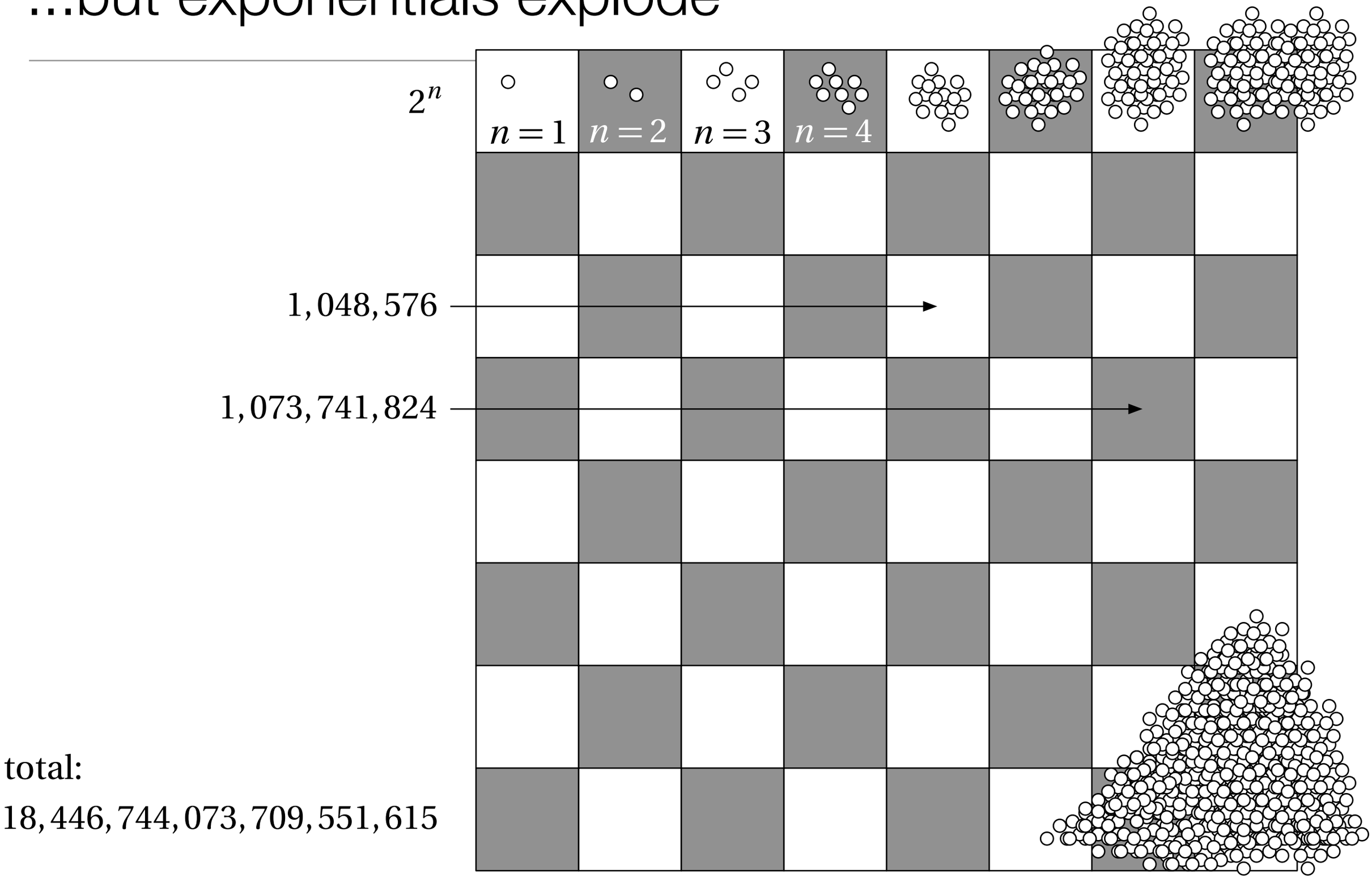
Polynomials don’t grow too badly as n grows...



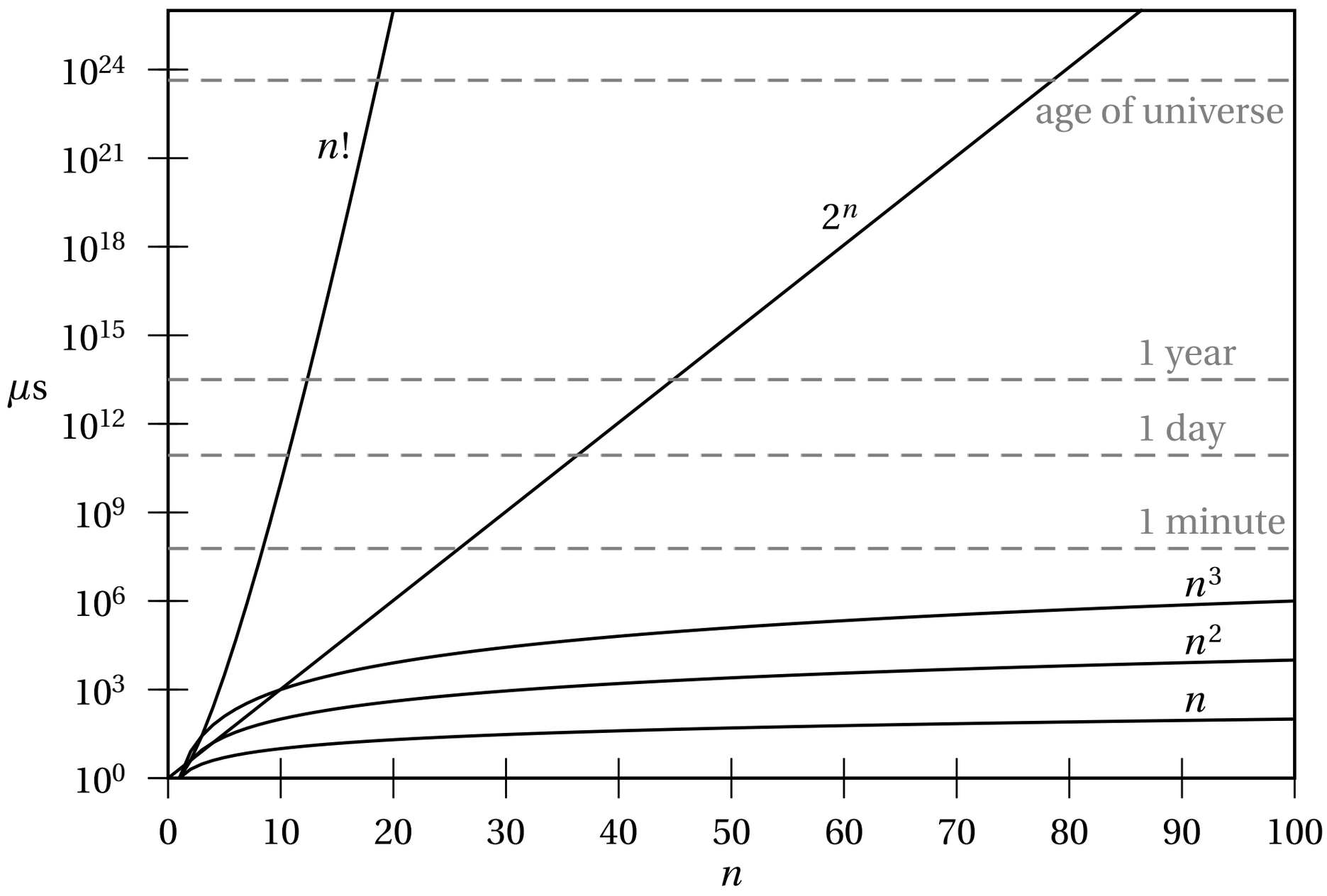
Polynomials don’t grow too badly as n grows...



...but exponentials explode



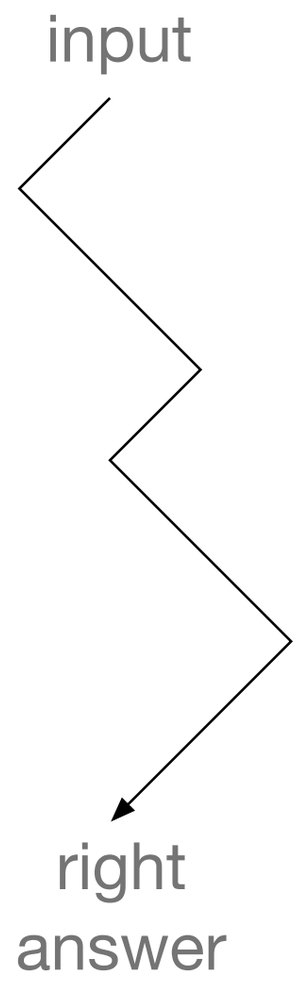
Until the end of the world



Chancy computations

Chancy computations

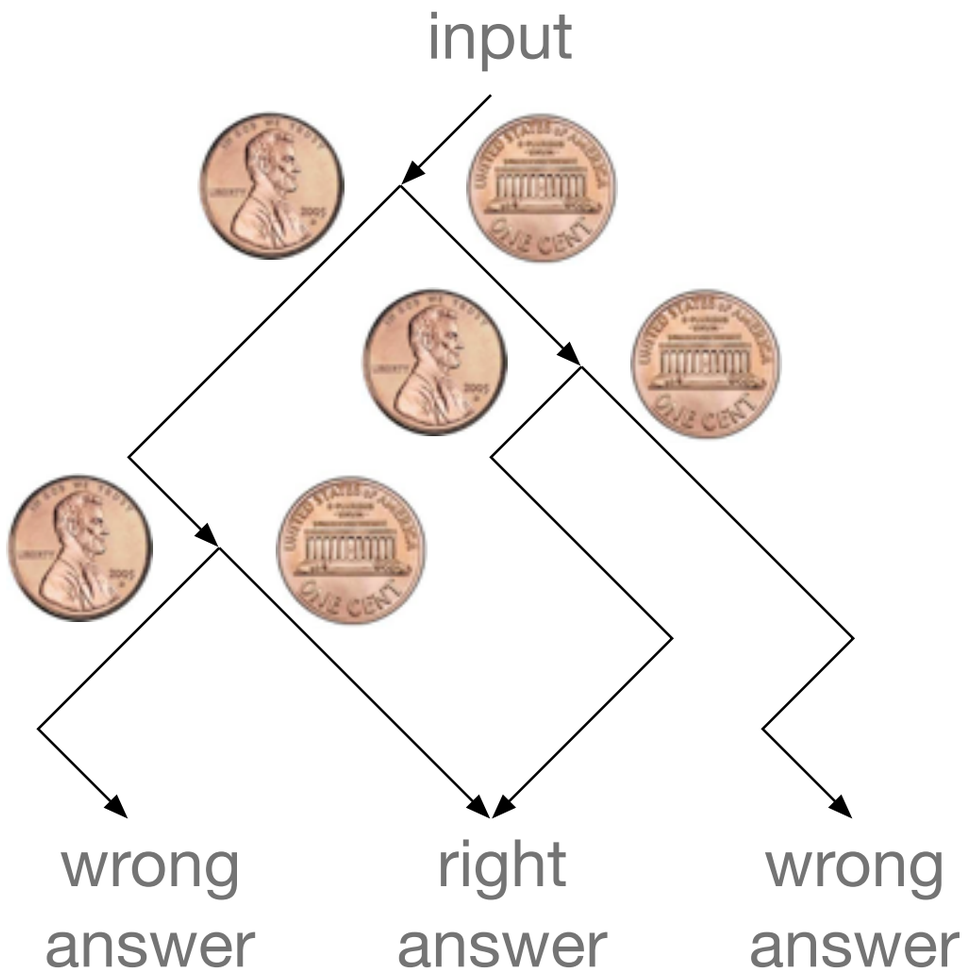
The usual kind of computation looks like this:



Chancy computations

The usual kind of computation looks like this:

A *randomized* computation looks like this:

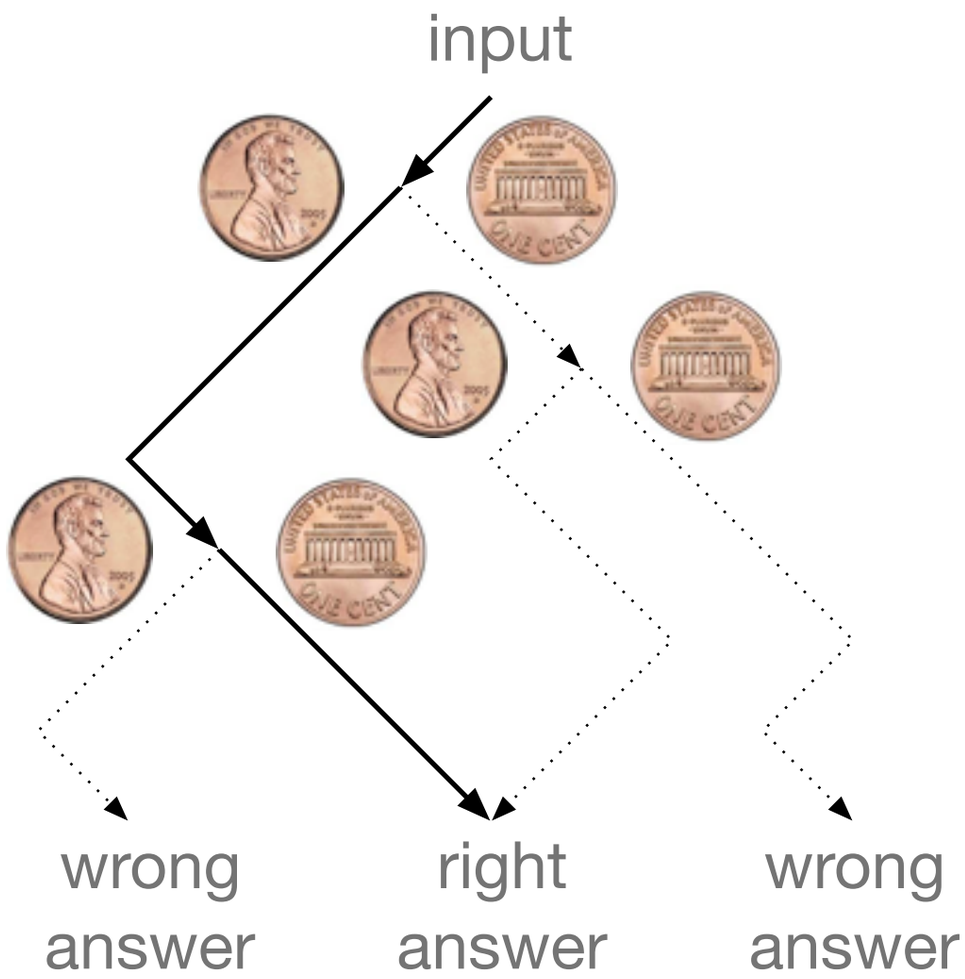


Chancy computations

The usual kind of computation looks like this:

A *randomized* computation looks like this:

Each path has a probability

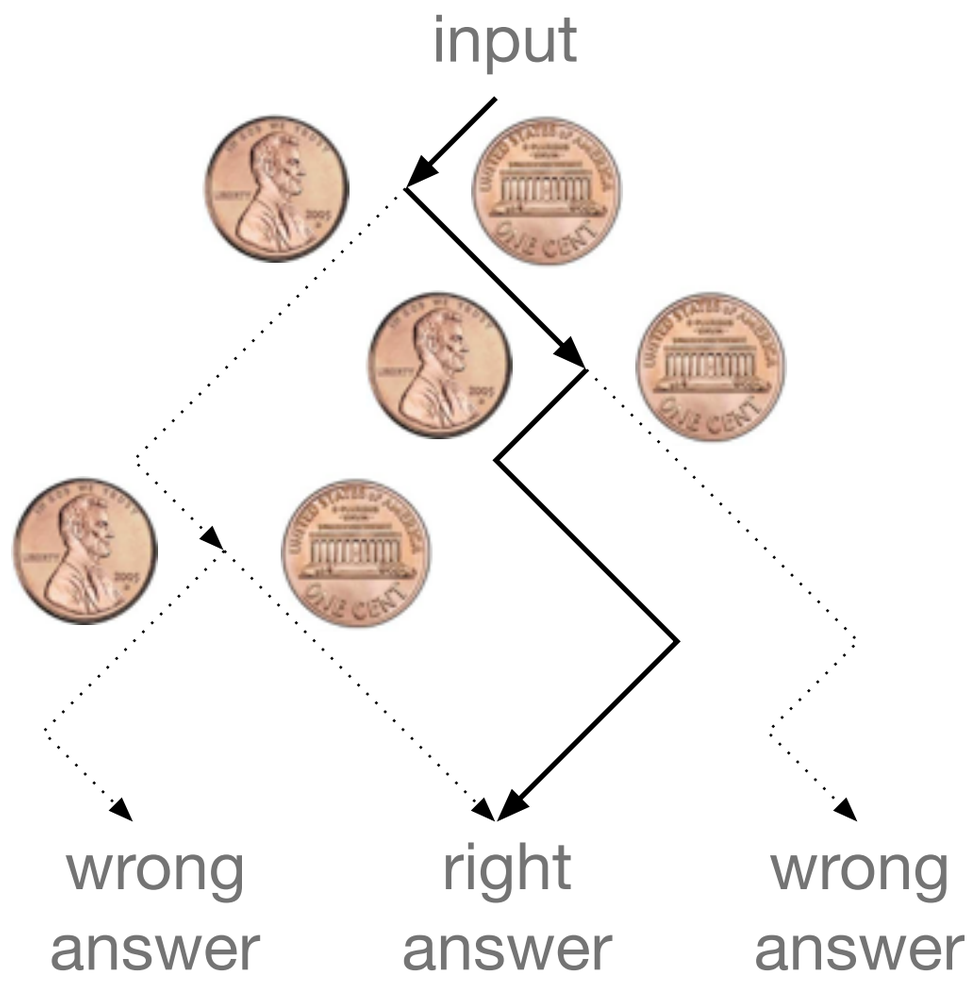


Chancy computations

The usual kind of computation looks like this:

A *randomized* computation looks like this:

Each path has a probability



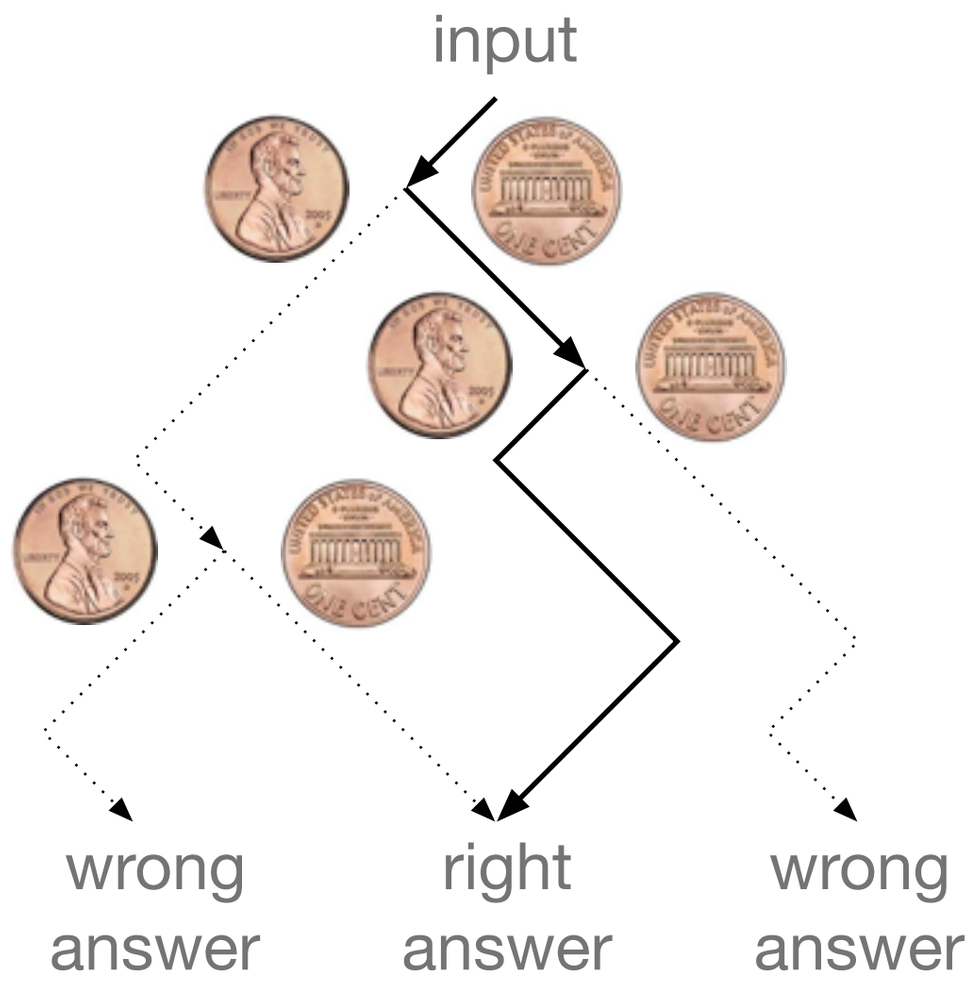
Chancy computations

The usual kind of computation looks like this:

A *randomized* computation looks like this:

Each path has a probability

The total probability of getting the right answer is the sum of the probabilities of the paths that lead there



Chancy computations

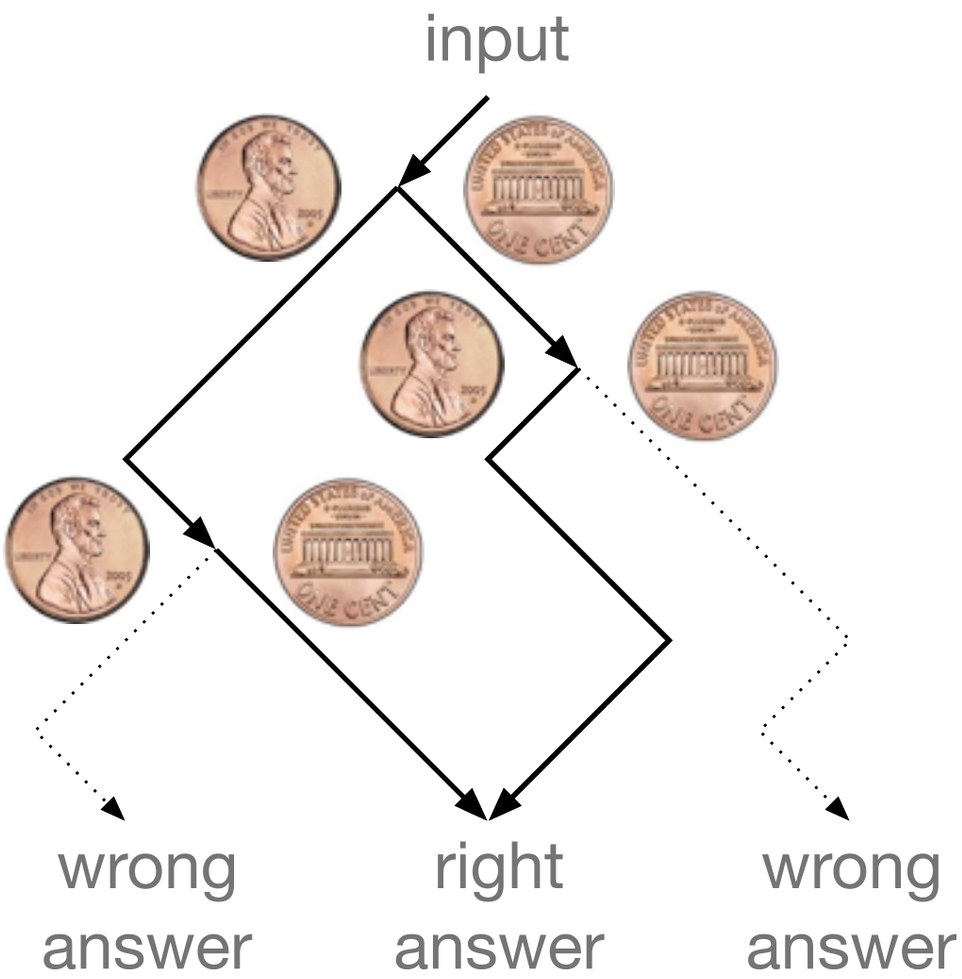
The usual kind of computation looks like this:

A *randomized* computation looks like this:

Each path has a probability

The total probability of getting the right answer is the sum of the probabilities of the paths that lead there

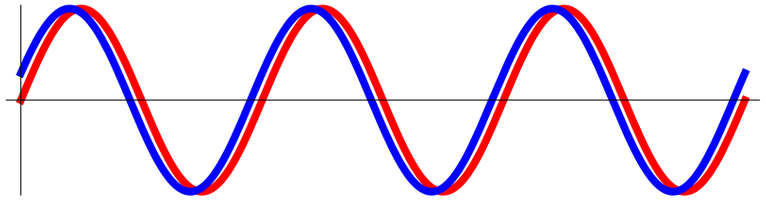
Some of our best algorithms work this way!
(Like telling whether or not a number is prime)

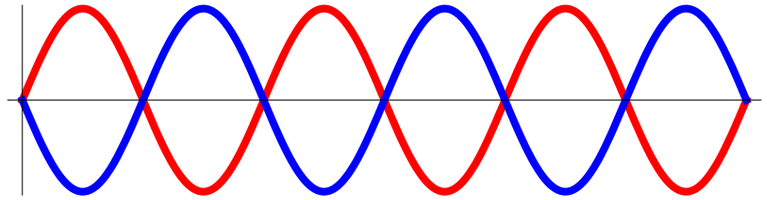


Enter the quantum

Probability is a number between 0 and 1; different possibilities add up

But waves have both *amplitude* and *phase*: they can add or cancel

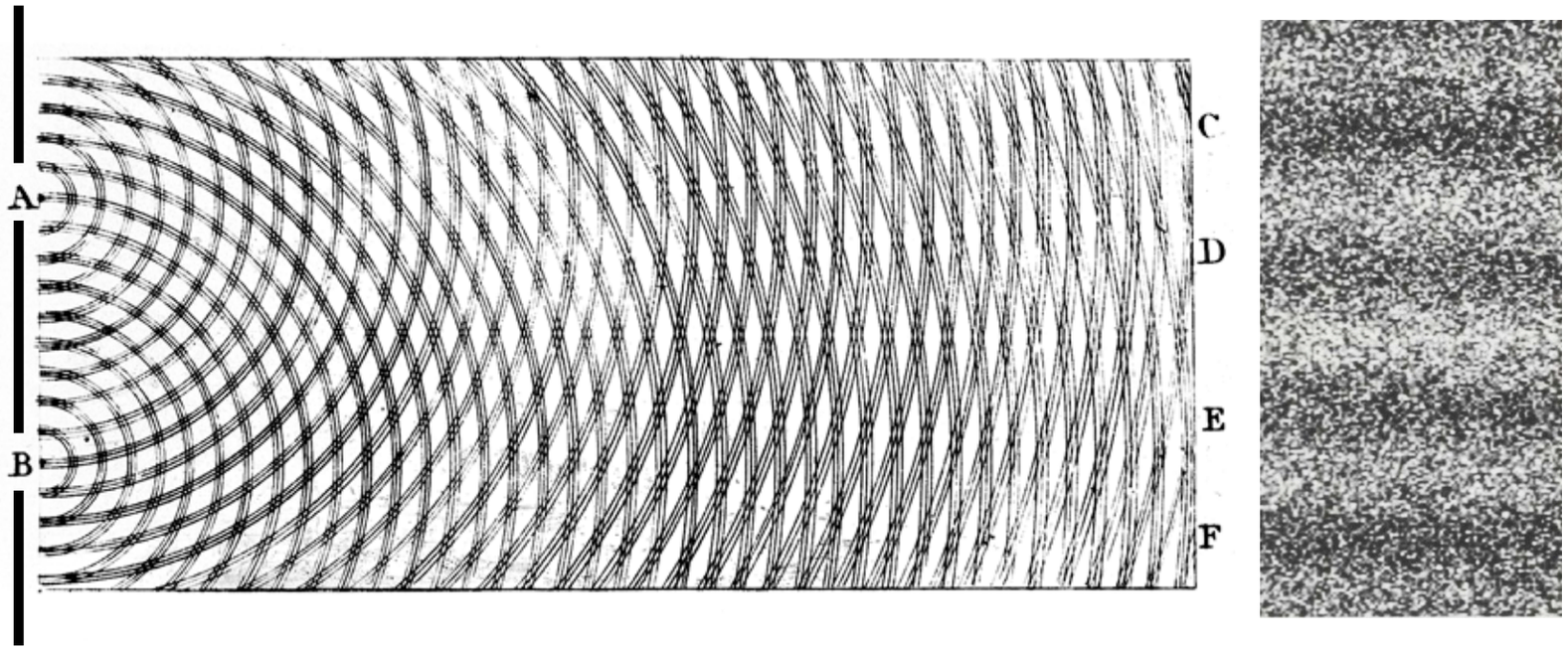




The two-slit experiment

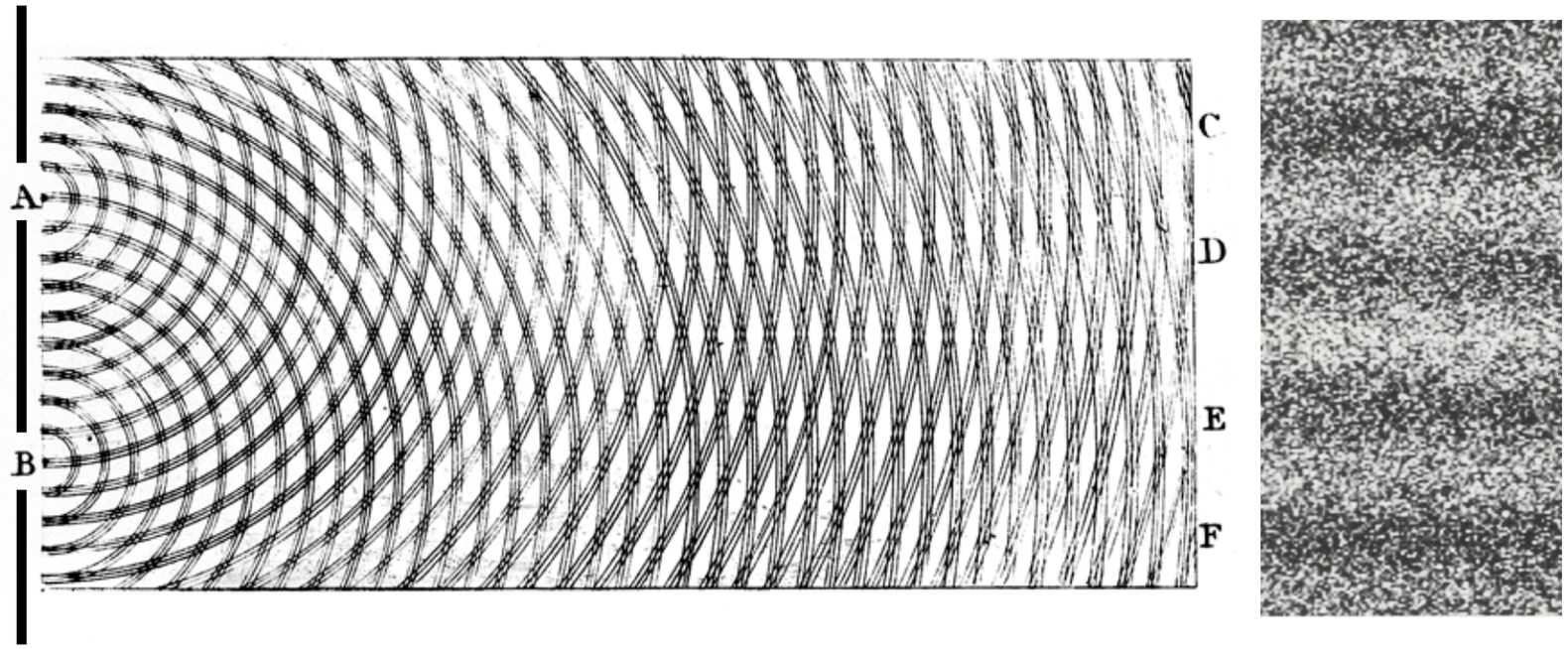
The two-slit experiment

Thomas Young, 1801: “light is possessed of opposite qualities, capable of neutralising or destroying each other.”



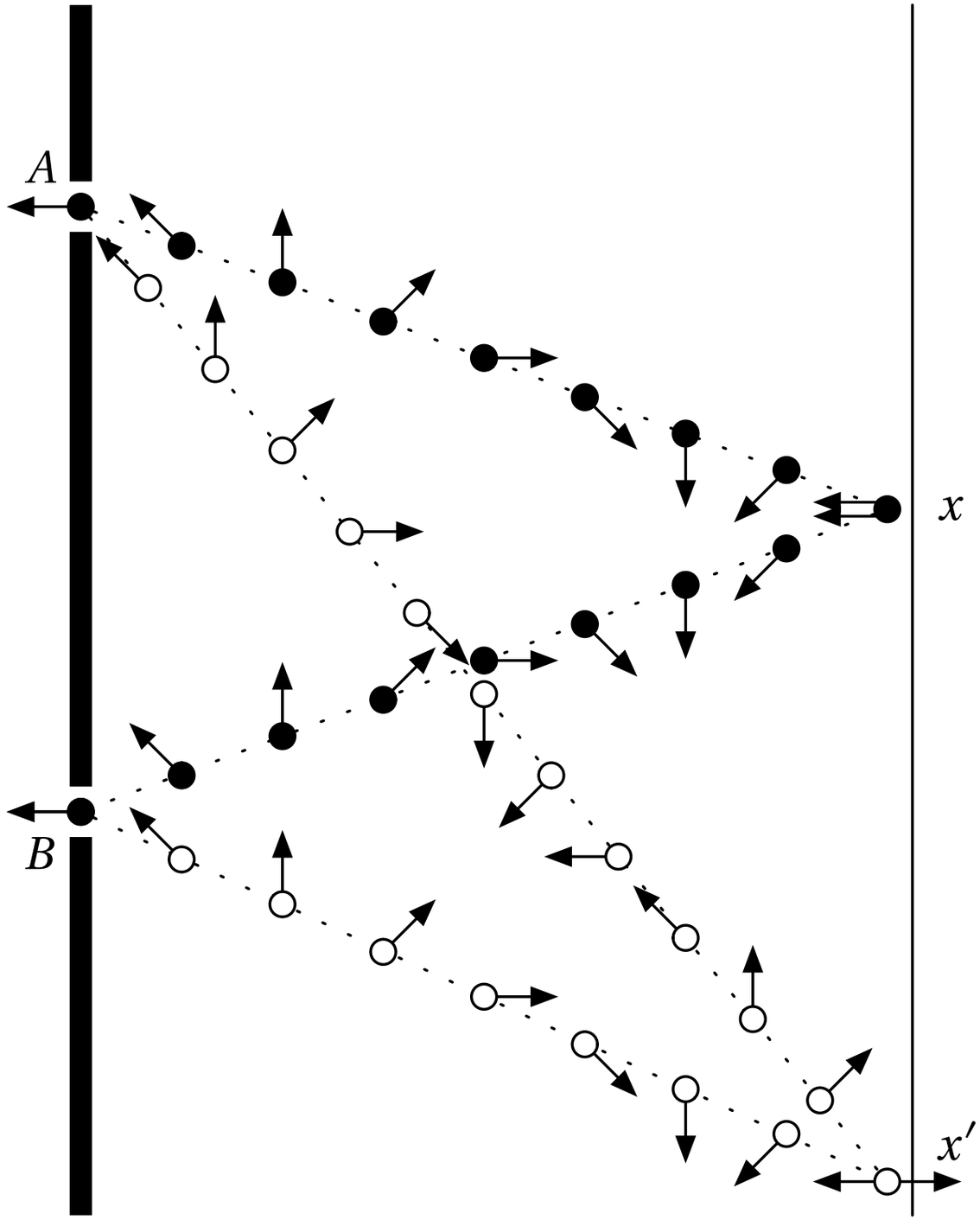
The two-slit experiment

Thomas Young, 1801: “light is possessed of opposite qualities, capable of neutralising or destroying each other.”



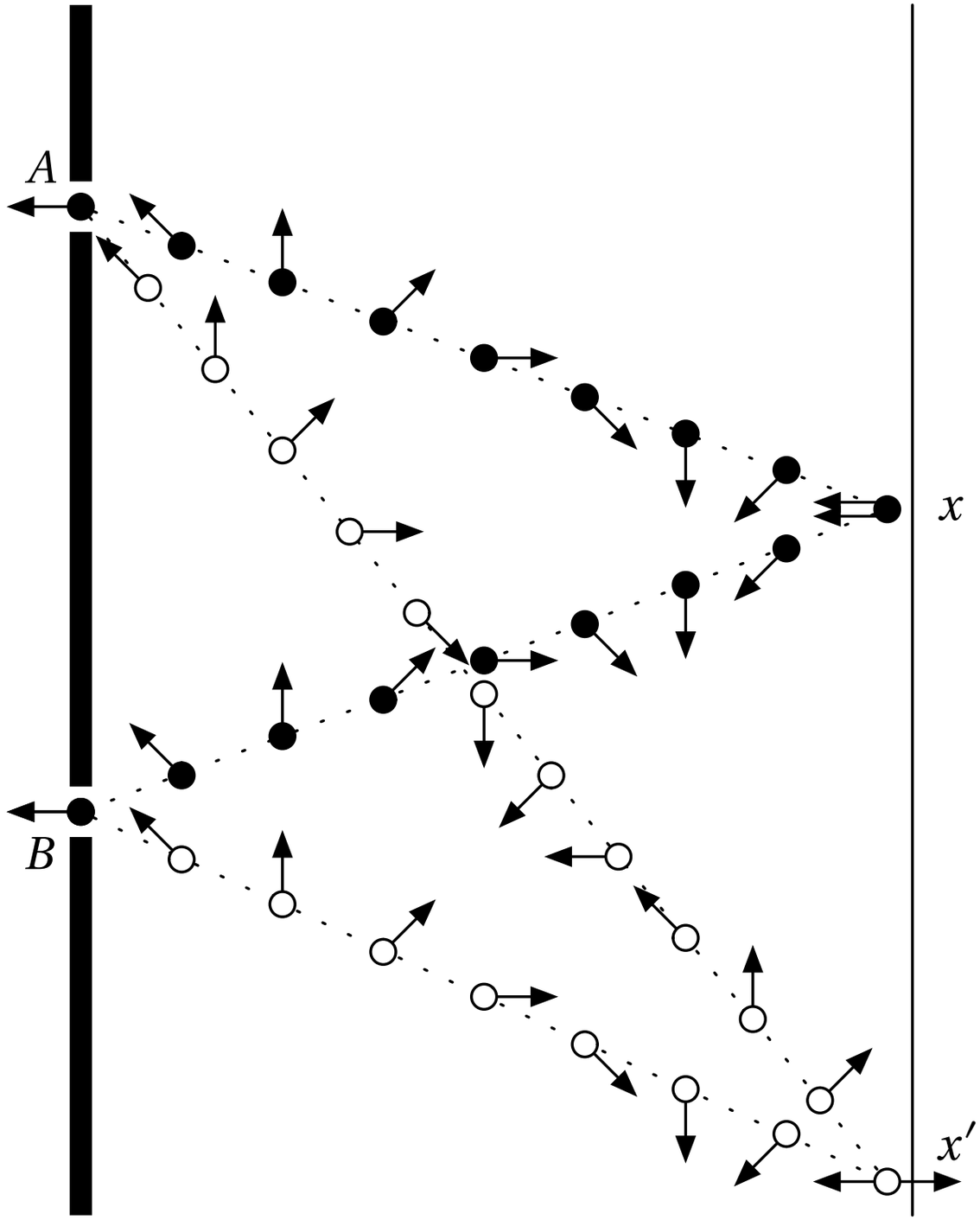
We see this even with one electron at a time! Even one particle is like a wave

In two places at once



In two places at once

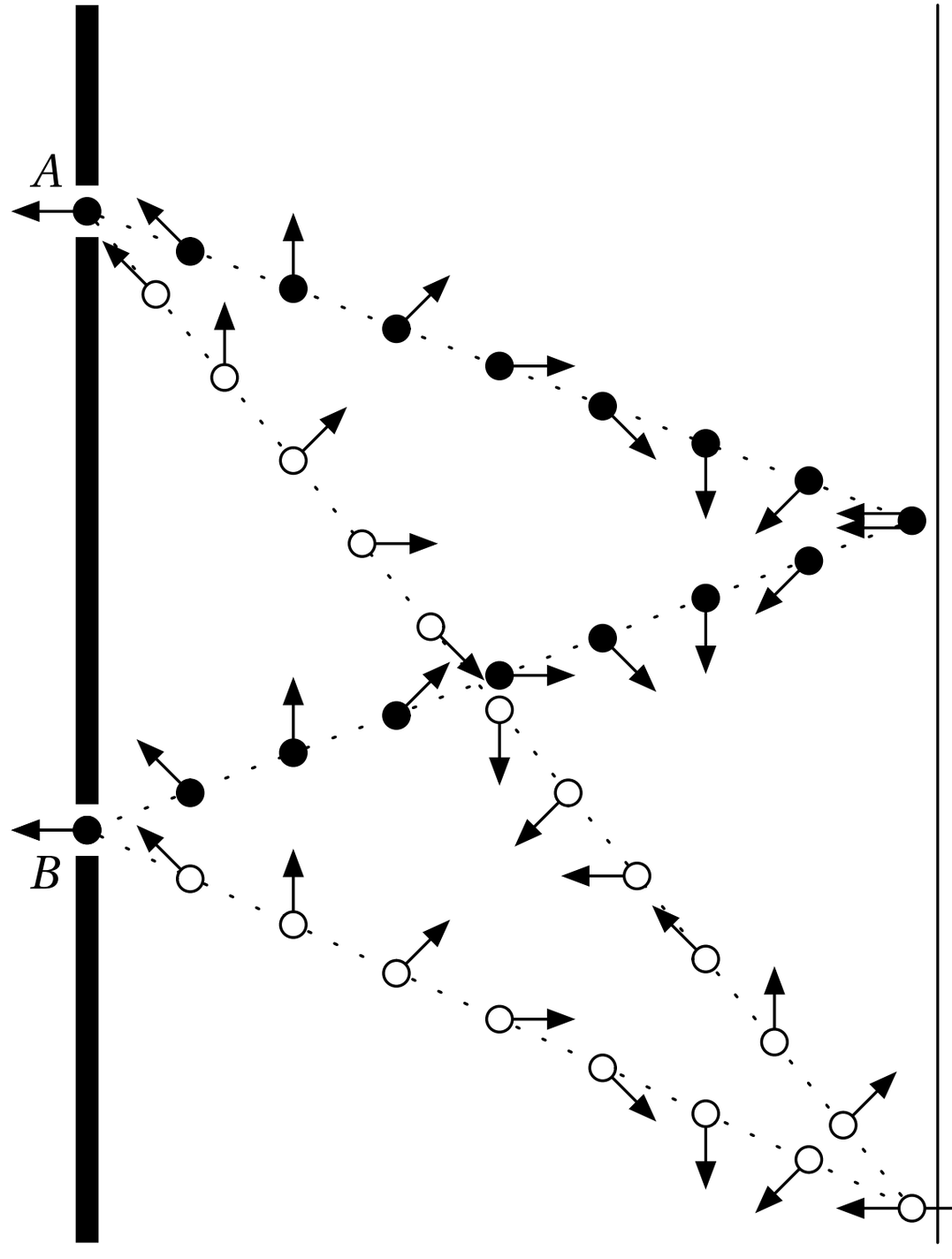
There are two paths the electron can take to each location



In two places at once

There are two paths the electron can take to each location

If these paths have the same “phase”, the paths add, and the probability is high

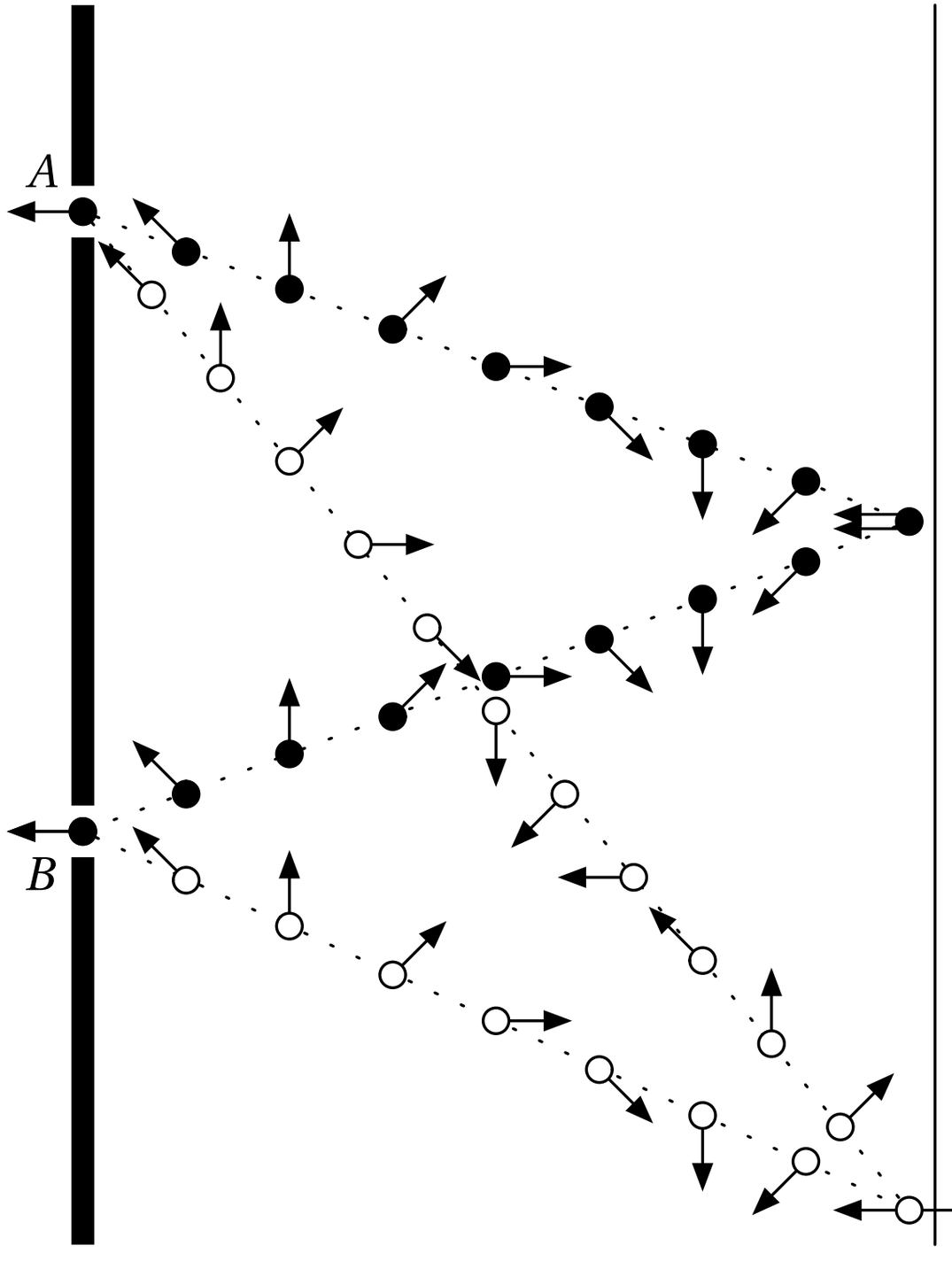


In two places at once

There are two paths the electron can take to each location

If these paths have the same “phase”, the paths add, and the probability is high

If they have opposite phase, they cancel



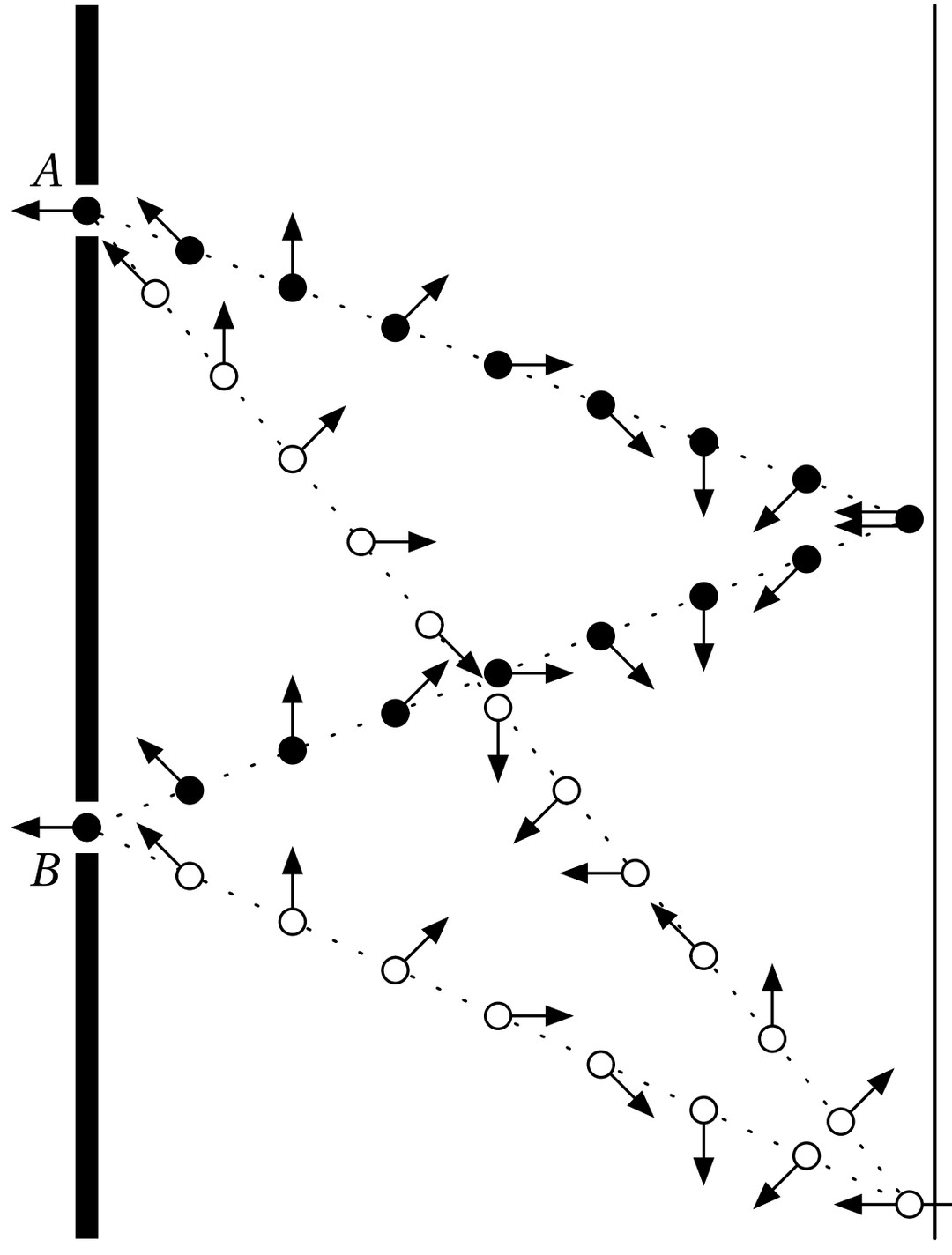
In two places at once

There are two paths the electron can take to each location

If these paths have the same “phase”, the paths add, and the probability is high

If they have opposite phase, they cancel

You can get to x' from *A* or *B*—and you never show up there at all!



In two places at once

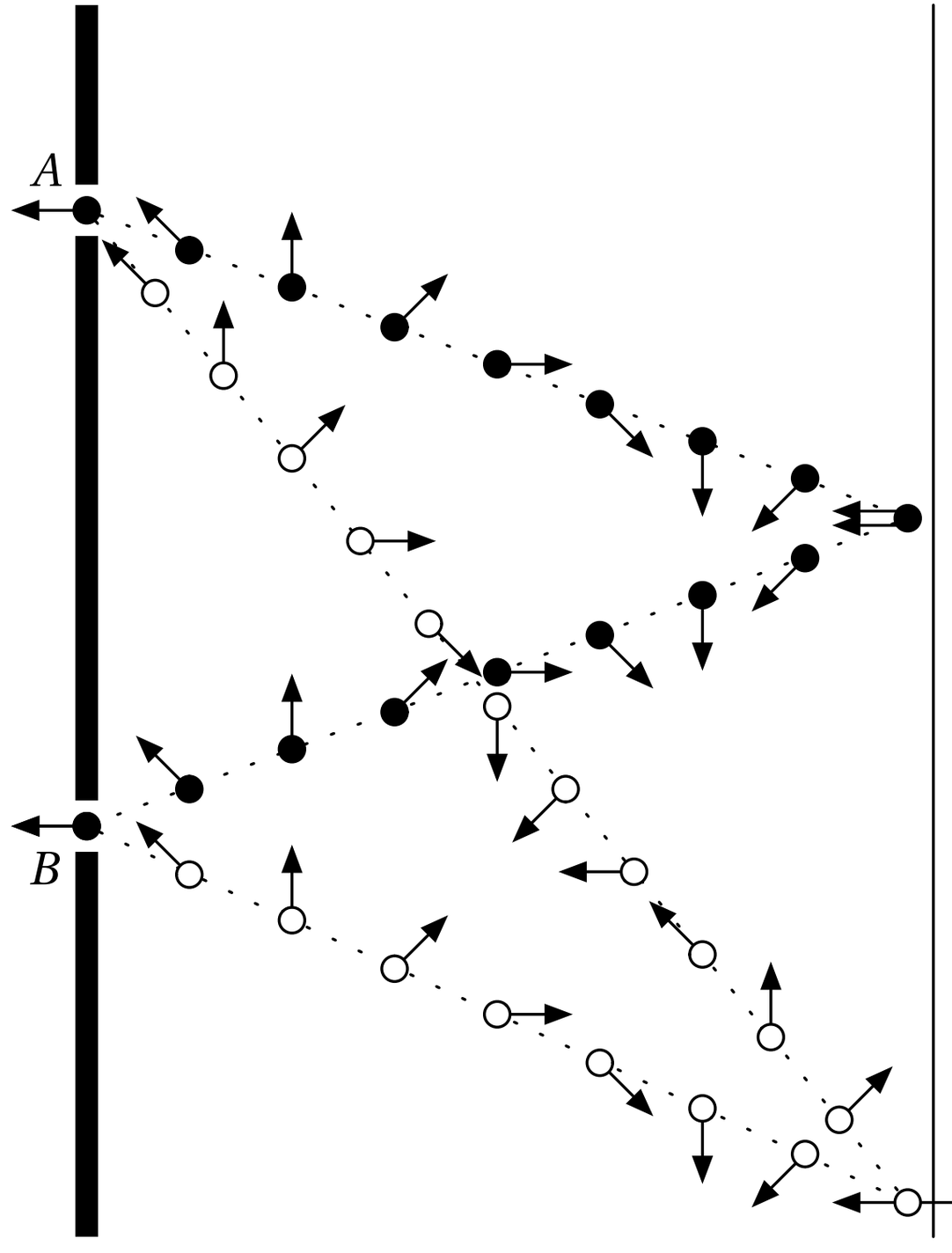
There are two paths the electron can take to each location

If these paths have the same “phase”, the paths add, and the probability is high

If they have opposite phase, they cancel

You can get to x' from *A* or *B*—and you never show up there at all!

Idea of quantum computation: probability adds up at the right answer, cancels out at the wrong ones



Shor’s factoring algorithm

Shor’s factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$2^4 = 15 + 1$$

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$16 - 1 = 15$$

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$16 - 1 = 15$$

$$4^2 - 1^2 = 15$$

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$16 - 1 = 15$$

$$4^2 - 1^2 = 15 \qquad\qquad x^2 - y^2 = (x - y)(x + y)$$

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$16 - 1 = 15$$

$$4^2 - 1^2 = 15 \qquad\qquad x^2 - y^2 = (x - y)(x + y)$$

$$(4 - 1) \times (4 + 1) = 15$$

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$16 - 1 = 15$$

$$4^2 - 1^2 = 15 \qquad\qquad x^2 - y^2 = (x - y)(x + y)$$

$$(4 - 1) \times (4 + 1) = 15$$

$$3 \times 5 = 15$$

Shor's factoring algorithm

Take the powers of some number x mod N, and see when this sequence repeats: 1, x, x^2, x^3...

For instance, to factor $N = 15$, look at the powers of 2:

t:	0	1	2	3	4	5	6	7	8
2^t:	1	2	4	8	1	2	4	8	1

$$16 - 1 = 15$$

$$4^2 - 1^2 = 15 \qquad x^2 - y^2 = (x - y)(x + y)$$

$$(4 - 1) \times (4 + 1) = 15$$

$$3 \times 5 = 15$$

But if the list of powers is very long, how can we tell when it repeats?

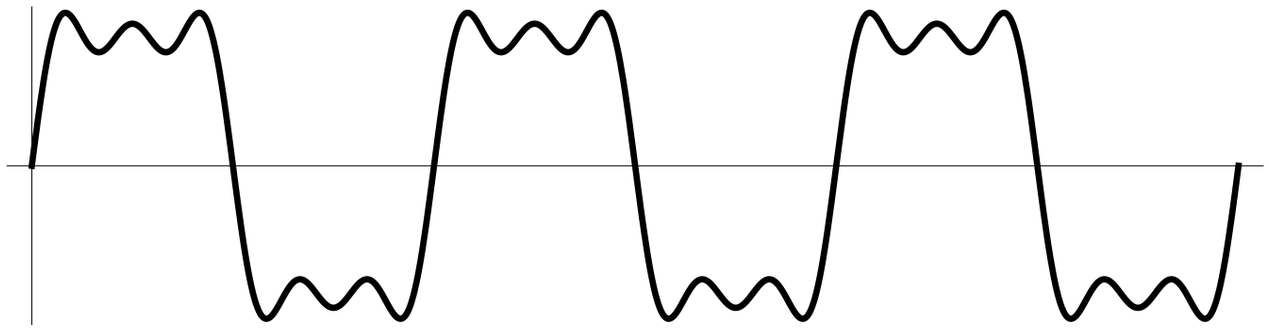
Hearing the harmonies

Hearing the harmonies

Fourier analysis breaks a signal down into a sum of frequencies

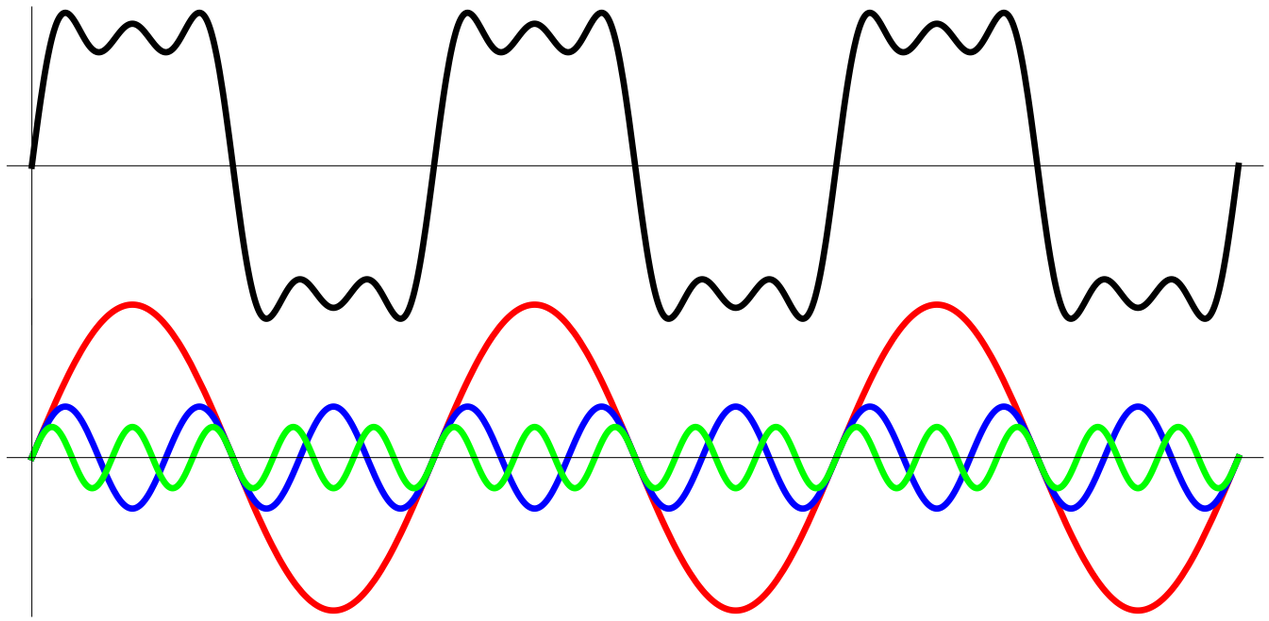
Hearing the harmonies

Fourier analysis breaks a signal down into a sum of frequencies



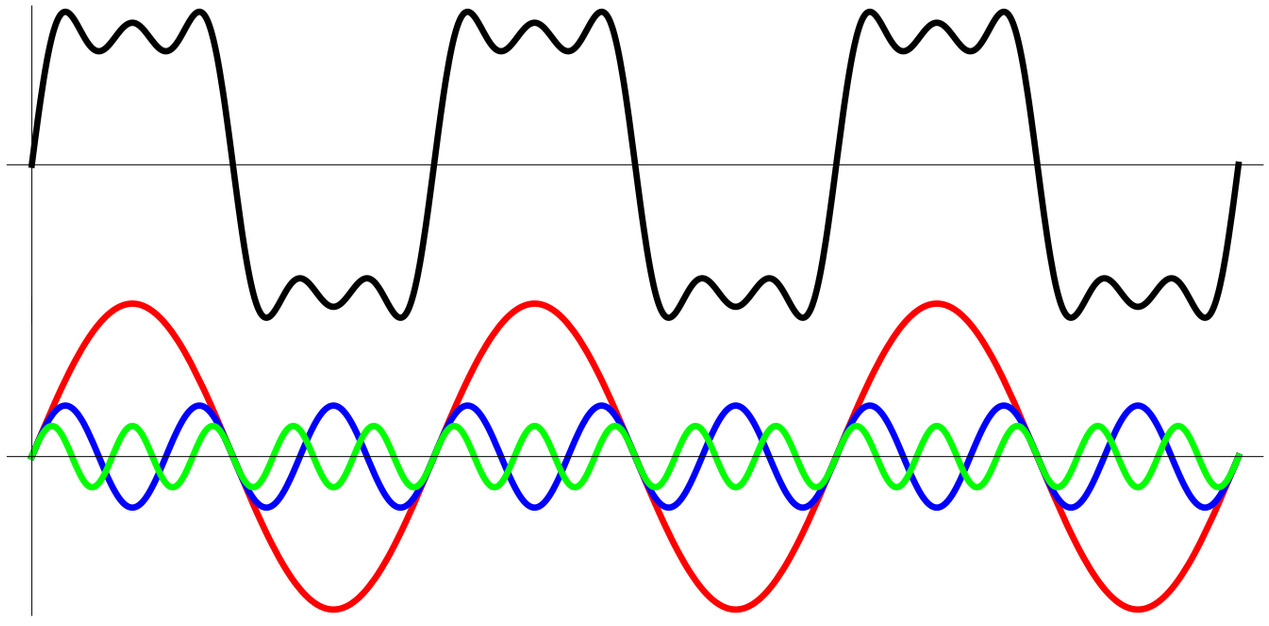
Hearing the harmonies

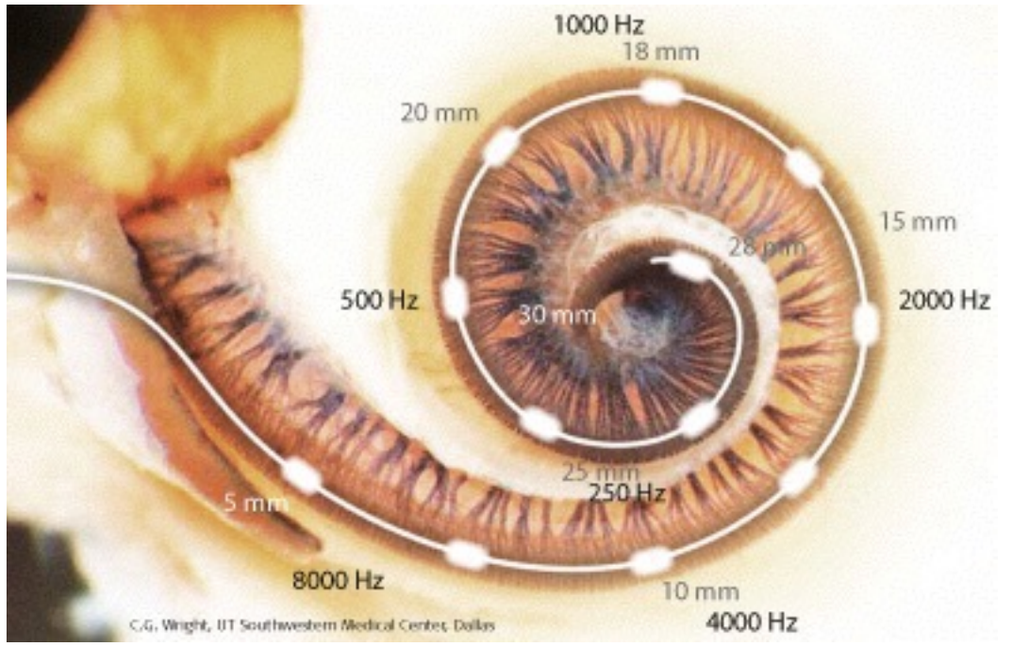
Fourier analysis breaks a signal down into a sum of frequencies



Hearing the harmonies

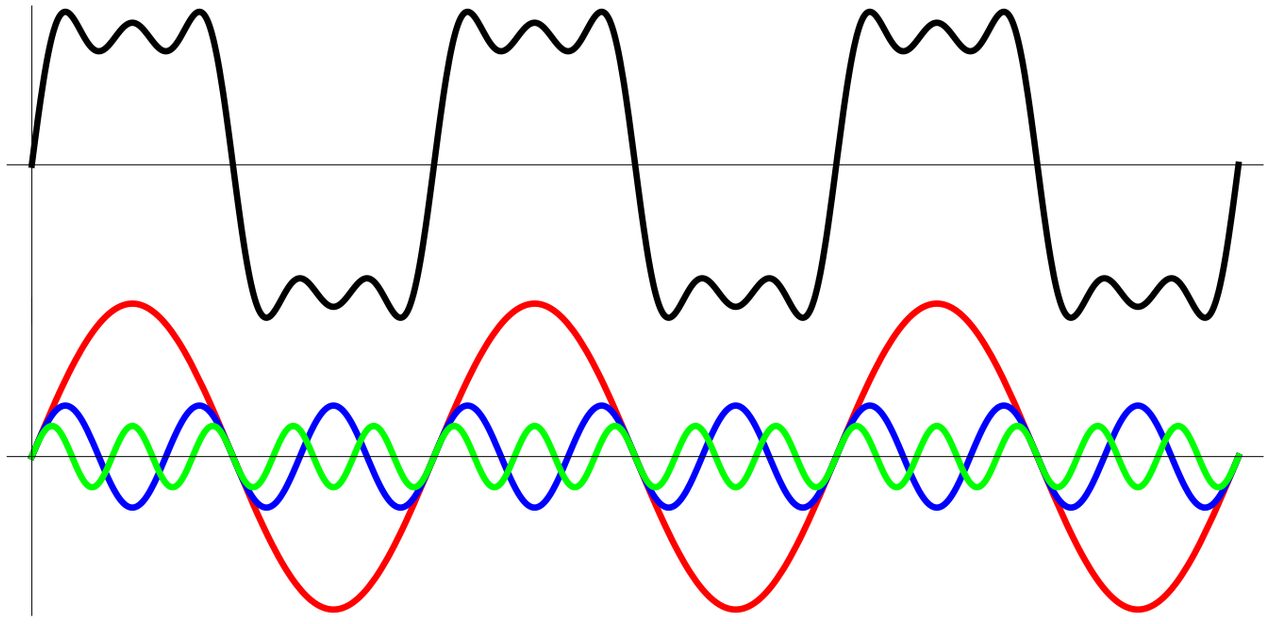
Fourier analysis breaks a signal down into a sum of frequencies

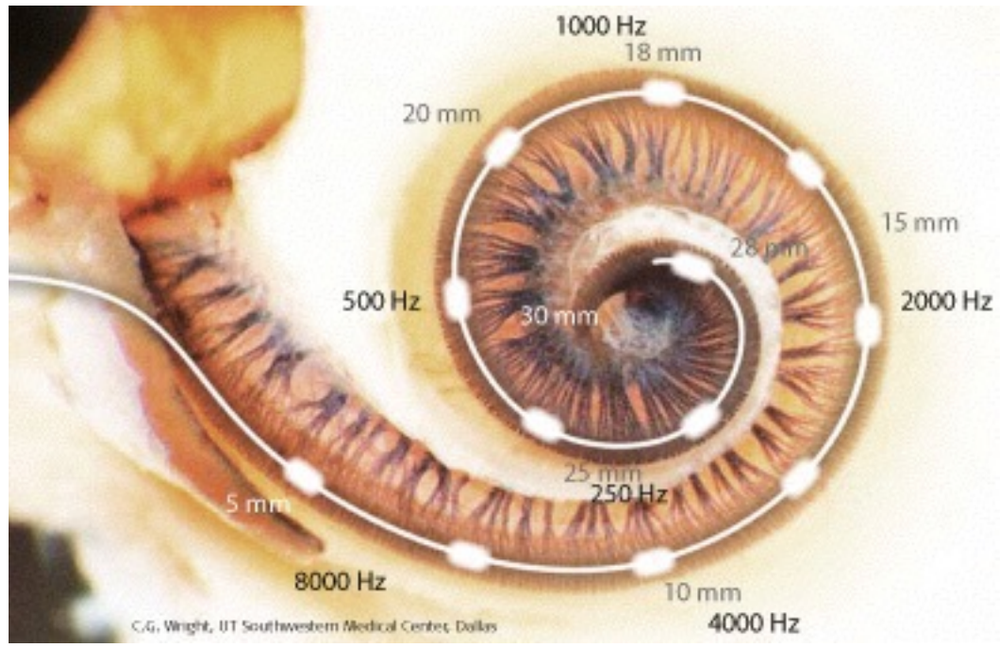




Hearing the harmonies

Fourier analysis breaks a signal down into a sum of frequencies

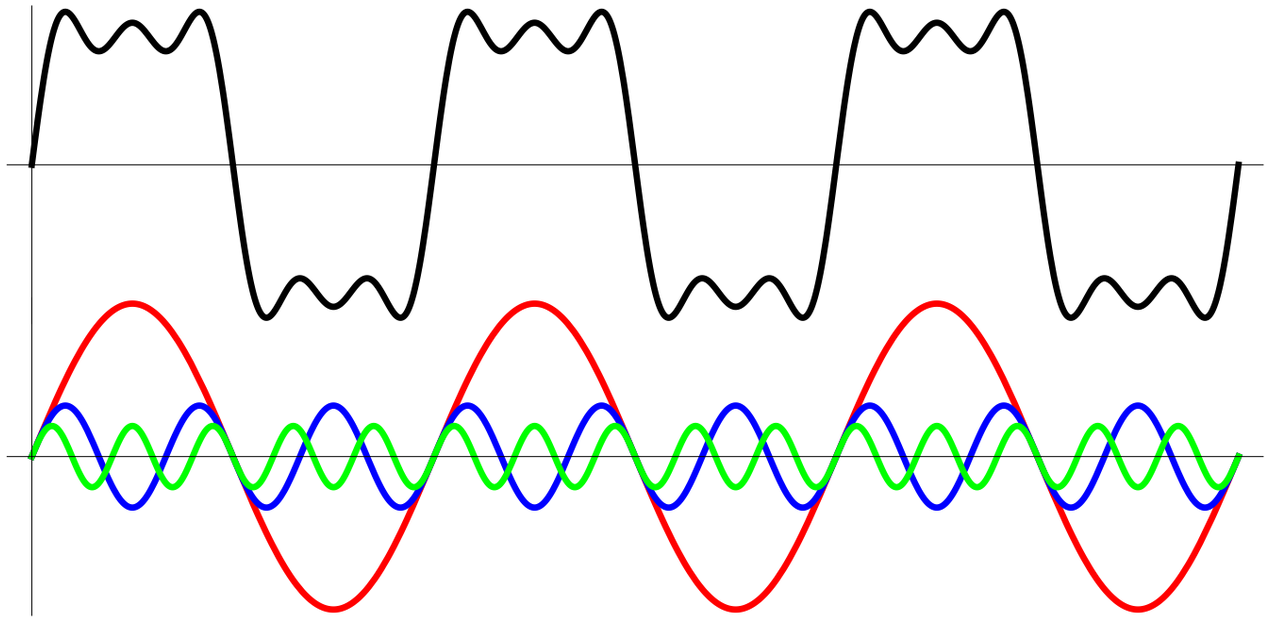


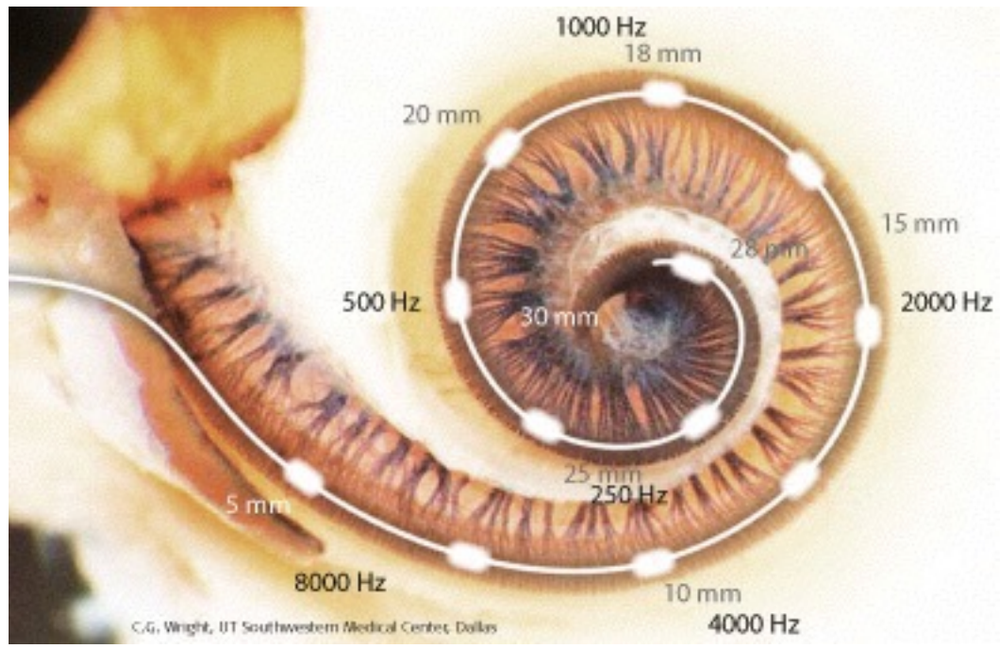


A quantum algorithm can pick out the frequencies in a sequence of numbers, even if it’s exponentially long!

Hearing the harmonies

Fourier analysis breaks a signal down into a sum of frequencies



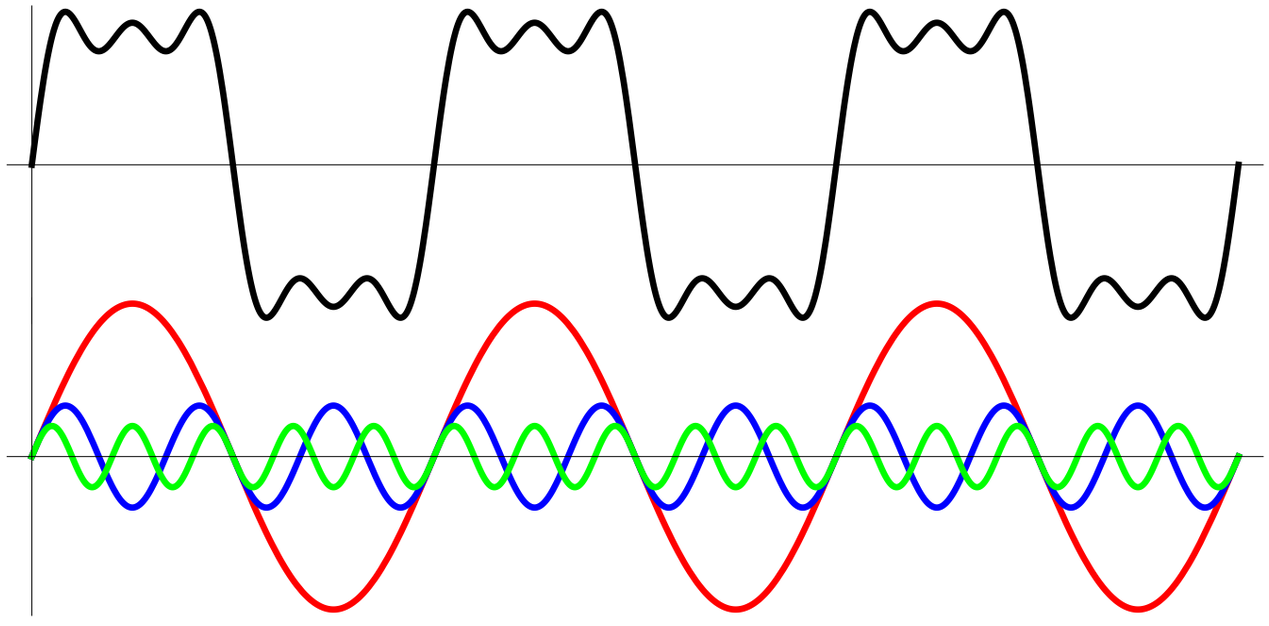


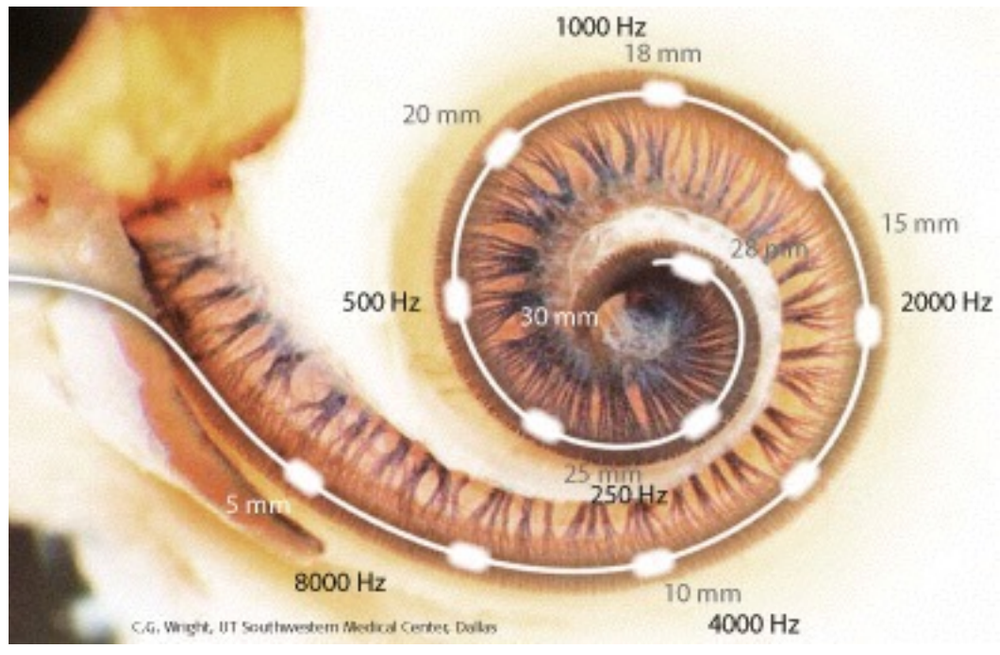
A quantum algorithm can pick out the frequencies in a sequence of numbers, even if it's exponentially long!

This means they can factor large numbers...

Hearing the harmonies

Fourier analysis breaks a signal down into a sum of frequencies





A quantum algorithm can pick out the frequencies in a sequence of numbers, even if it's exponentially long!

This means they can factor large numbers...

and break RSA cryptography

What next?

What next?

Are there cryptosystems that even quantum computers can’t break?

What next?

Are there cryptosystems that even quantum computers can’t break?

Are there problems that even quantum computers can’t solve?

Error-correcting codes

Error-correcting codes

A set of possible messages, with large differences between every pair of possibilities

Error-correcting codes

A set of possible messages, with large differences between every pair of possibilities

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Error-correcting codes

A set of possible messages, with large differences between every pair of possibilities

$$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Error-correcting codes

A set of possible messages, with large differences between every pair of possibilities

$$\begin{bmatrix}0\\1\\1\end{bmatrix} \quad \begin{bmatrix}0 & 0 & 0 & 1 & 1 & 1 & 1\\0 & 1 & 1 & 0 & 0 & 1 & 1\\1 & 0 & 1 & 0 & 1 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 1 & 0 & 0 & 1 & 1 & 0\end{bmatrix}$$

Error-correcting codes

A set of possible messages, with large differences between every pair of possibilities

$$\begin{bmatrix}0\\1\\1\end{bmatrix} \quad \begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}$$

$$\begin{bmatrix}1&1&0&0&1&1&0\end{bmatrix}$$

Every combination is at least 4 bits different from every other one:
if any one bit gets flipped, we can fix it

Error-correcting codes

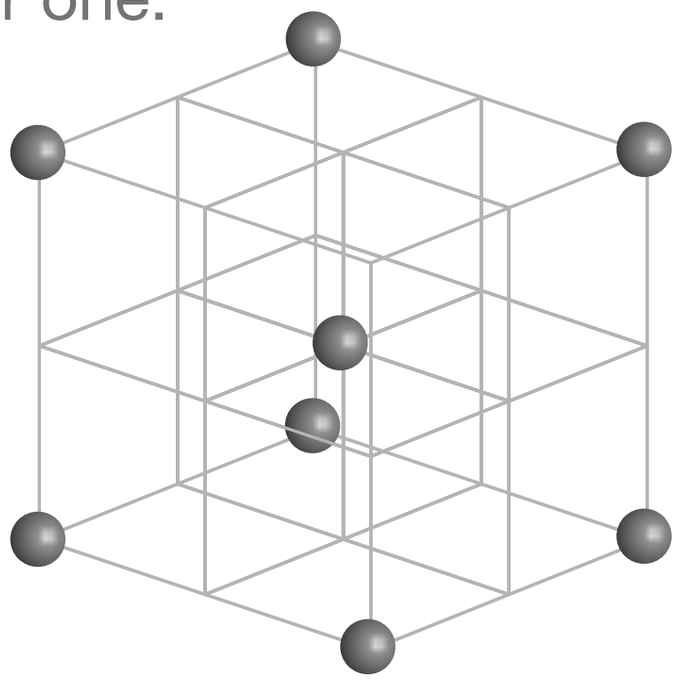
A set of possible messages, with large differences between every pair of possibilities

$$\begin{bmatrix}0\\1\\1\end{bmatrix} \quad \begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}$$

$$\begin{bmatrix}1&1&0&0&1&1&0\end{bmatrix}$$

Every combination is at least 4 bits different from every other one: if any one bit gets flipped, we can fix it

A crystal of codewords: in high dimensions, finding the closest one is NP-hard—a needle in a haystack



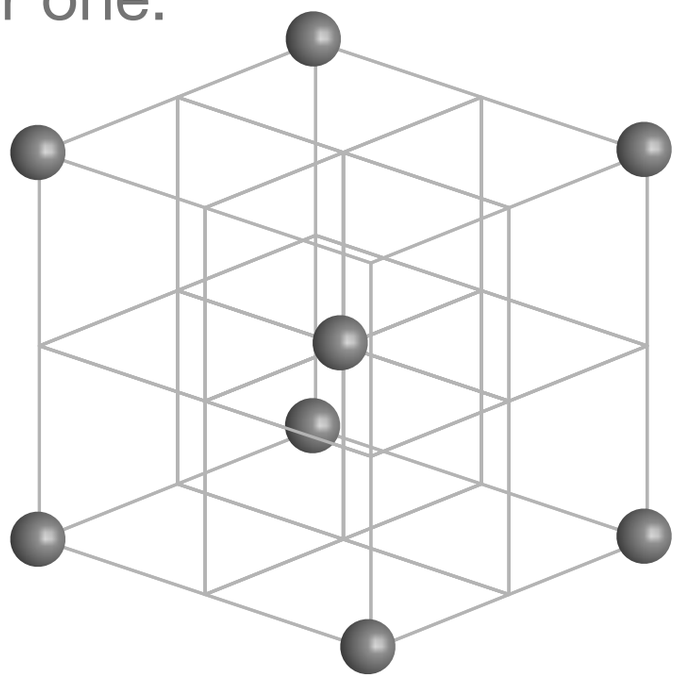
Error-correcting codes

A set of possible messages, with large differences between every pair of possibilities

$$\begin{bmatrix}0\\1\\1\end{bmatrix} \quad \begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}$$

$$\begin{bmatrix}1&1&0&0&1&1&0\end{bmatrix}$$

Every combination is at least 4 bits different from every other one: if any one bit gets flipped, we can fix it



A crystal of codewords: in high dimensions, finding the closest one is NP-hard—a needle in a haystack

But for some special codes, we can do this quickly

The McEliece cryptosystem

The McEliece cryptosystem

Alice has an error-correcting code in which she can correct errors easily

The McEliece cryptosystem

Alice has an error-correcting code in which she can correct errors easily

She publishes a scrambled version of it, twisting and turning the crystal in order to make it look unfamiliar

The McEliece cryptosystem

Alice has an error-correcting code in which she can correct errors easily

She publishes a scrambled version of it, twisting and turning the crystal in order to make it look unfamiliar

Bob composes his message in the scrambled code, adds some noise, and sends it

The McEliece cryptosystem

Alice has an error-correcting code in which she can correct errors easily

She publishes a scrambled version of it, twisting and turning the crystal in order to make it look unfamiliar

Bob composes his message in the scrambled code, adds some noise, and sends it

Removing Bob’s noise is hard for an eavesdropper (we hope), but easy for Alice

Can we unscramble the code?

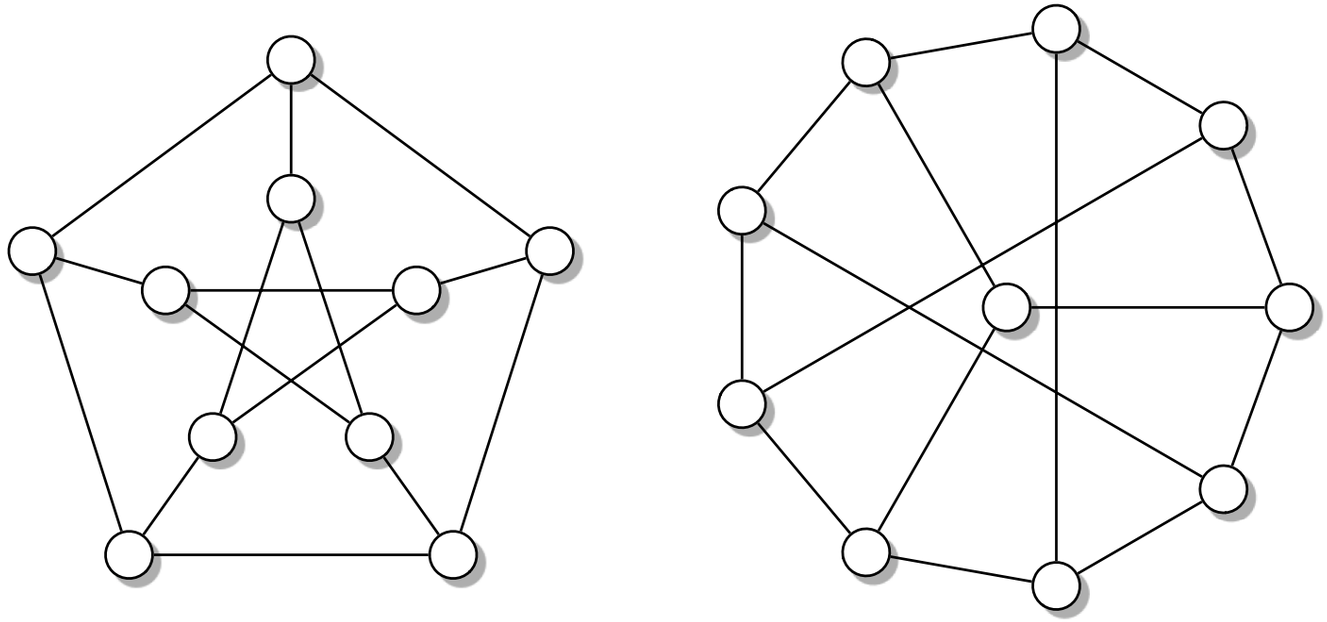
Can we unscramble the code?

Even if we know Alice’s internal code, how can we tell how she scrambled it?

Can we unscramble the code?

Even if we know Alice’s internal code, how can we tell how she scrambled it?

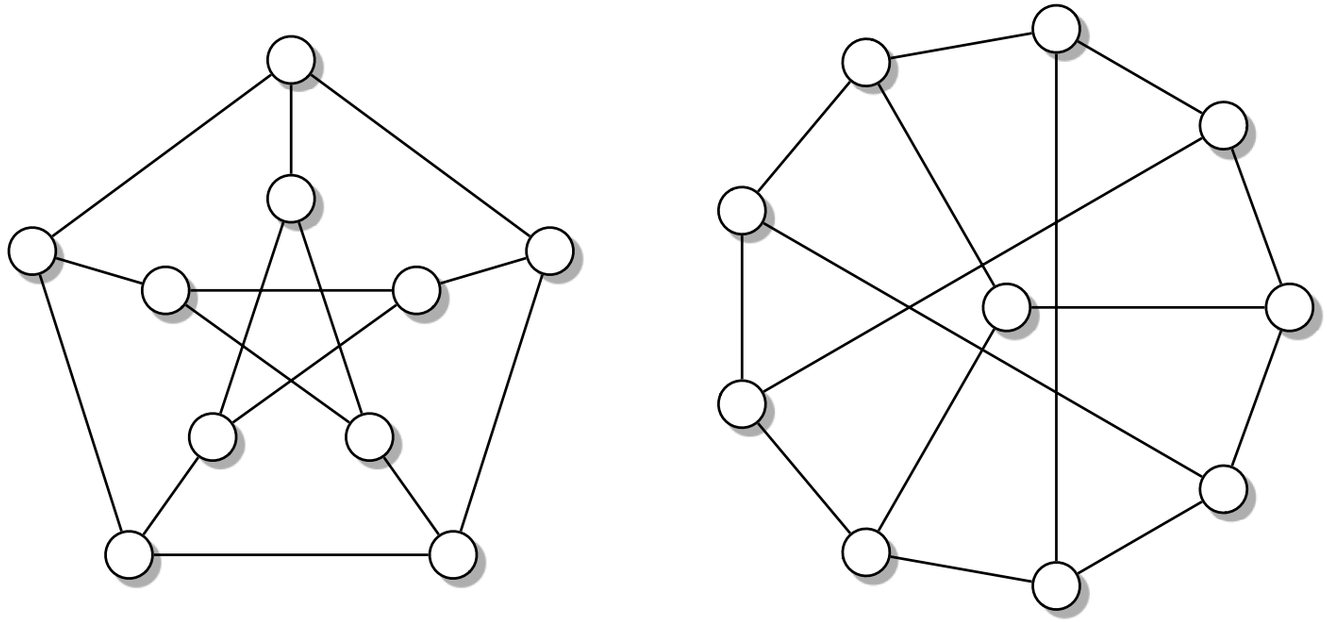
Similar: can I rearrange these nodes to turn one network into the other?



Can we unscramble the code?

Even if we know Alice’s internal code, how can we tell how she scrambled it?

Similar: can I rearrange these nodes to turn one network into the other?



We don’t know how to solve this problem on classical computers…
What about quantum ones?

Fearful symmetry

Fearful symmetry

Emmy Noether: physics is about symmetry

Fearful symmetry

Emmy Noether: physics is about symmetry

Einstein: symmetries of space-time are not what Galileo thought they were

Fearful symmetry

Emmy Noether: physics is about symmetry

Einstein: symmetries of space-time are not what Galileo thought they were

Periodicity is symmetry: shift and it stays the same

Fearful symmetry

Emmy Noether: physics is about symmetry

Einstein: symmetries of space-time are not what Galileo thought they were

Periodicity is symmetry: shift and it stays the same

Fearful symmetry

Emmy Noether: physics is about symmetry

Einstein: symmetries of space-time are not what Galileo thought they were

Periodicity is symmetry: shift and it stays the same

Higher-dimensional symmetries:

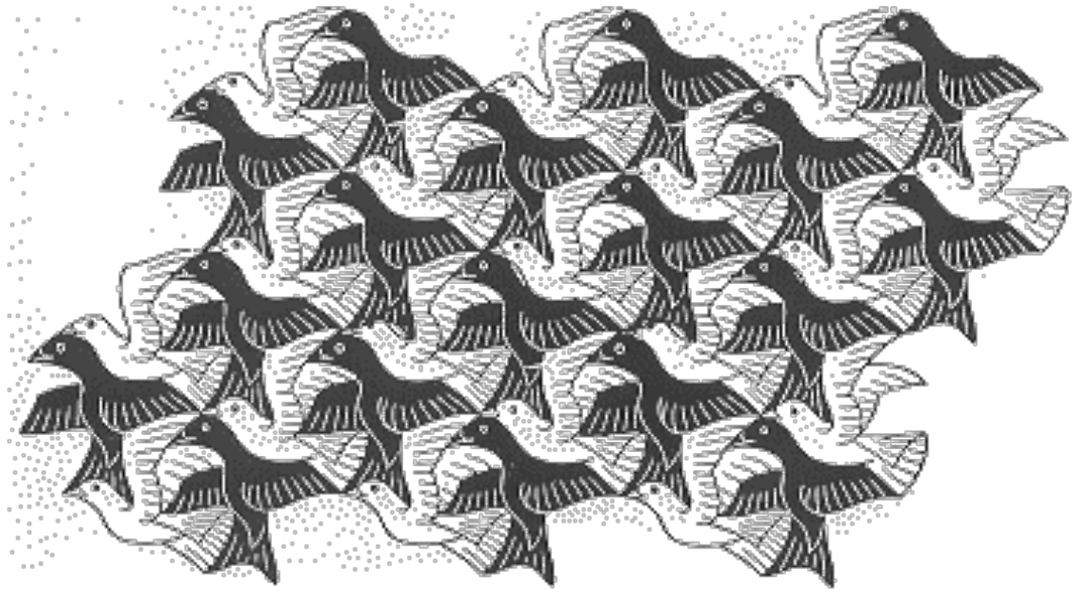
Fearful symmetry

Emmy Noether: physics is about symmetry

Einstein: symmetries of space-time are not what Galileo thought they were

Periodicity is symmetry: shift and it stays the same

Higher-dimensional symmetries:



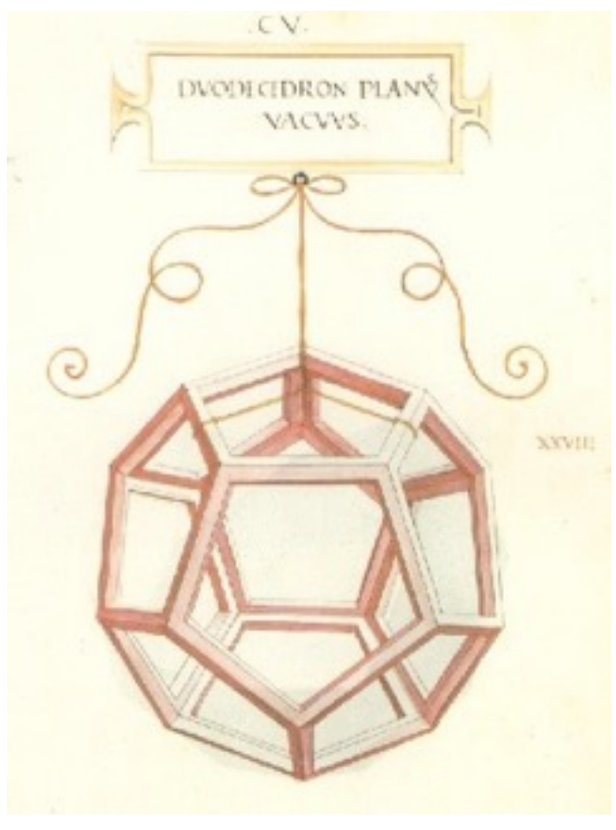
Fearful symmetry

Emmy Noether: physics is about symmetry

Einstein: symmetries of space-time are not what Galileo thought they were

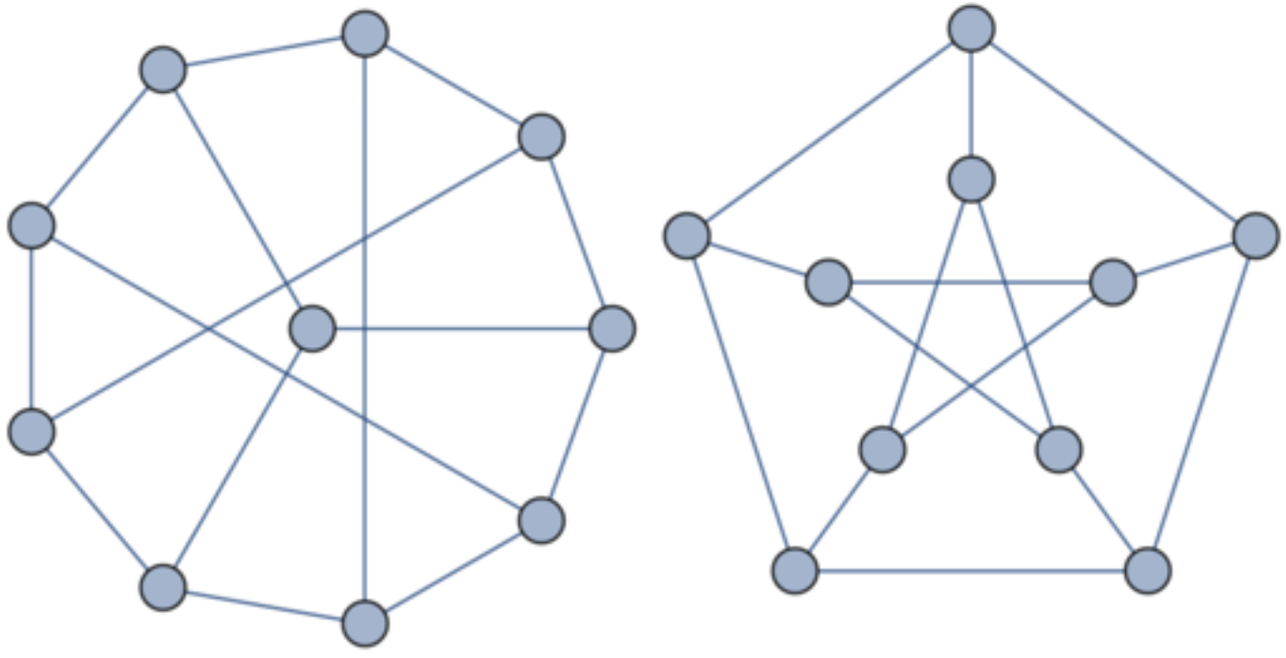
Periodicity is symmetry: shift and it stays the same

Higher-dimensional symmetries:



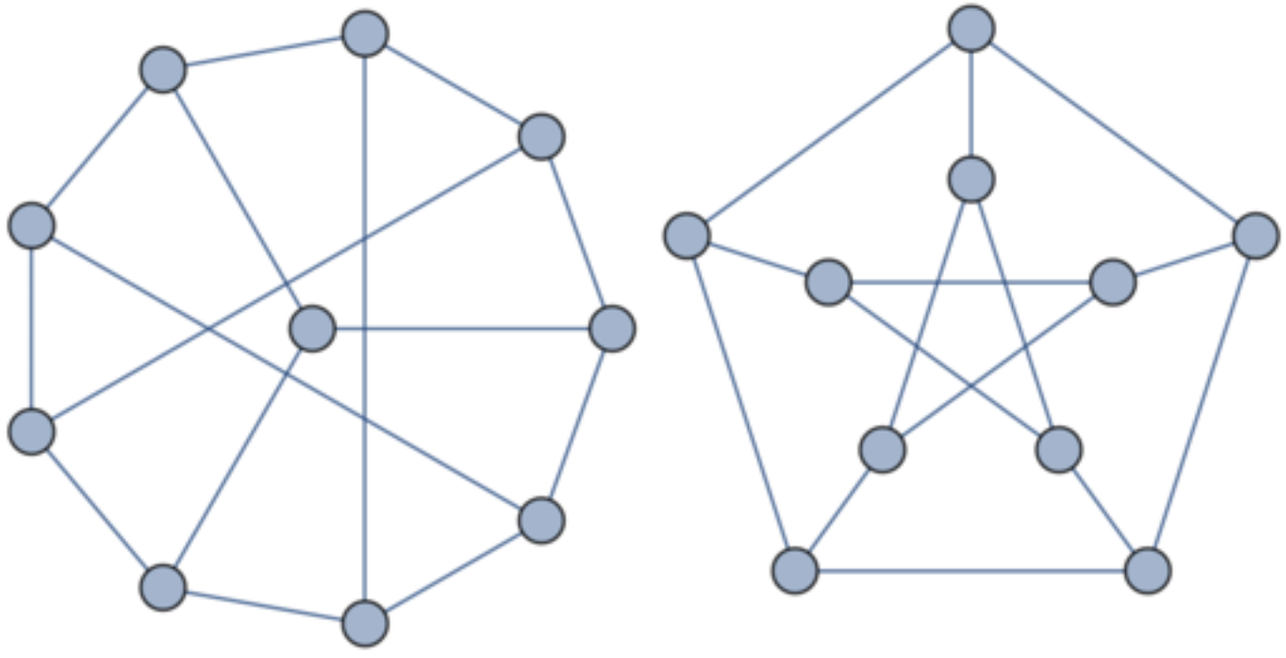
Fearful symmetry

Can quantum computers detect the symmetries of these graphs —
the rearrangements that turn them into each other?

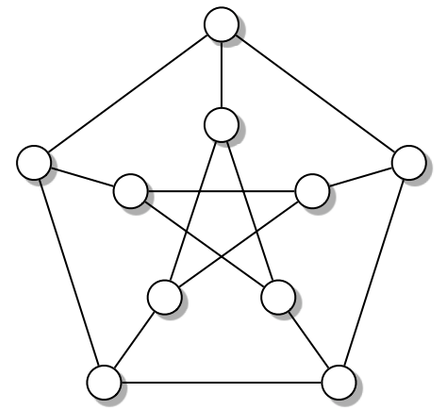


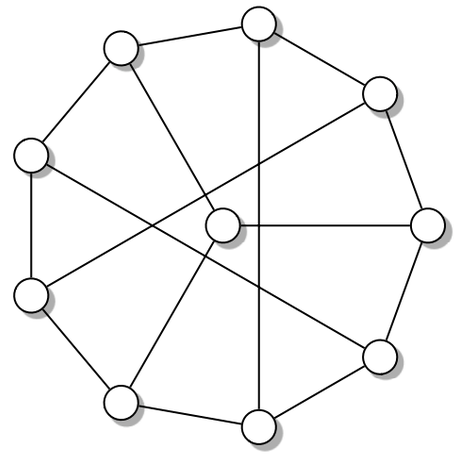
Fearful symmetry

Can quantum computers detect the symmetries of these graphs —
the rearrangements that turn them into each other?

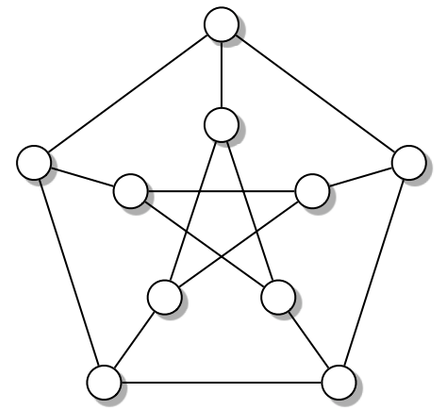


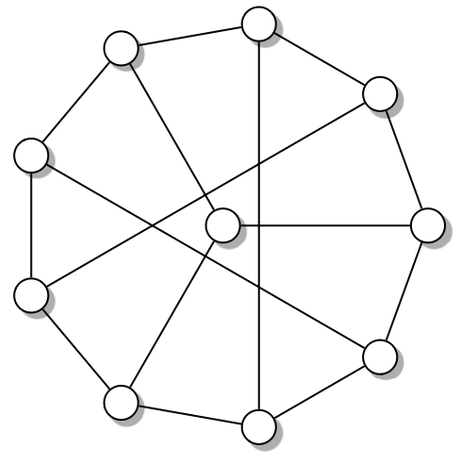
Scramblings and symmetries





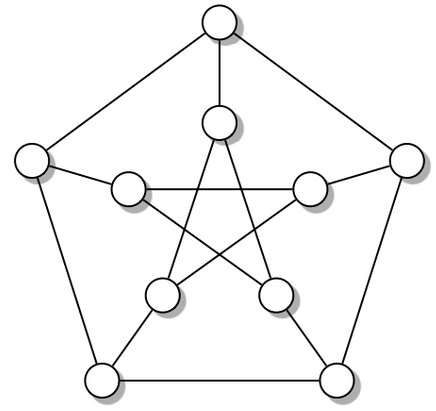
Scramblings and symmetries

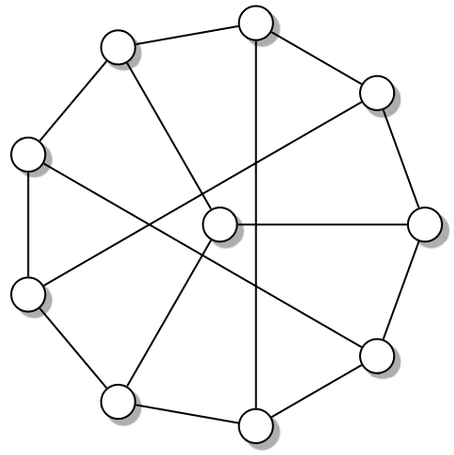




Scramblings of codes or networks are *permutations*

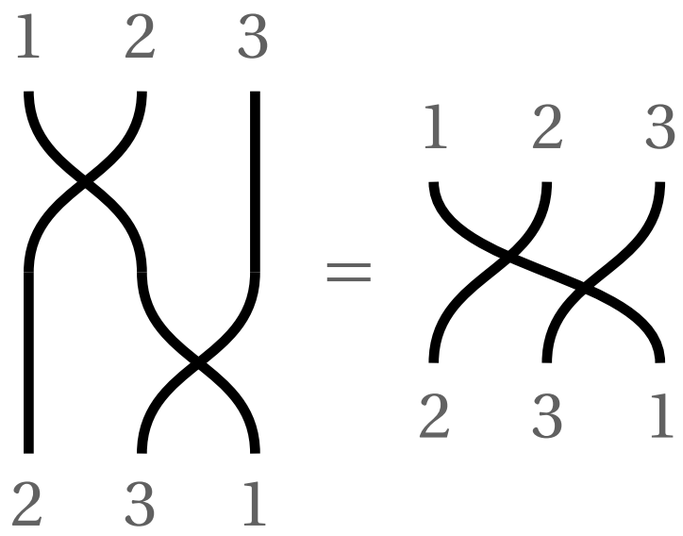
Scramblings and symmetries





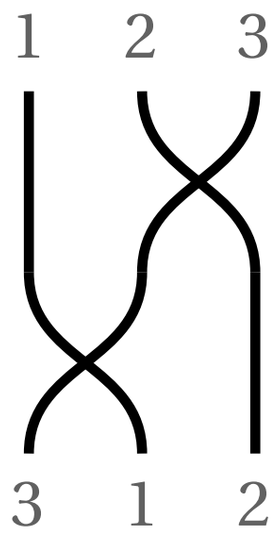
Scramblings of codes or networks are *permutations*

We can “multiply” them, but *xy* ≠ *yx*

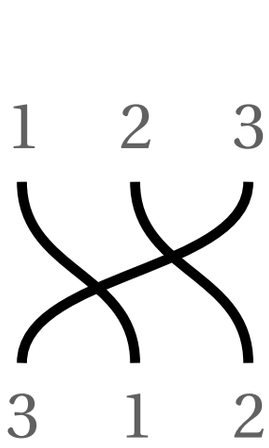


=

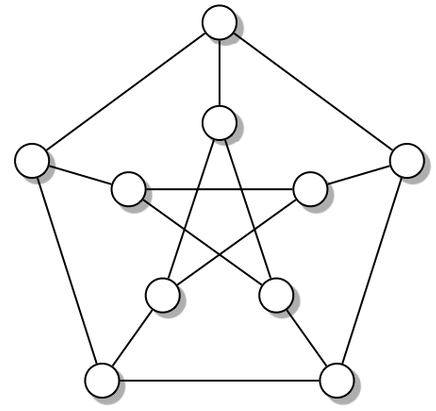
but

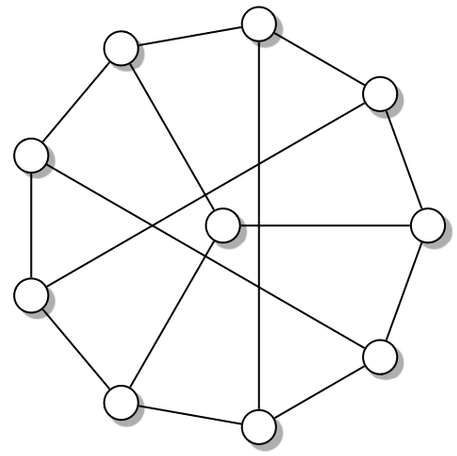


=



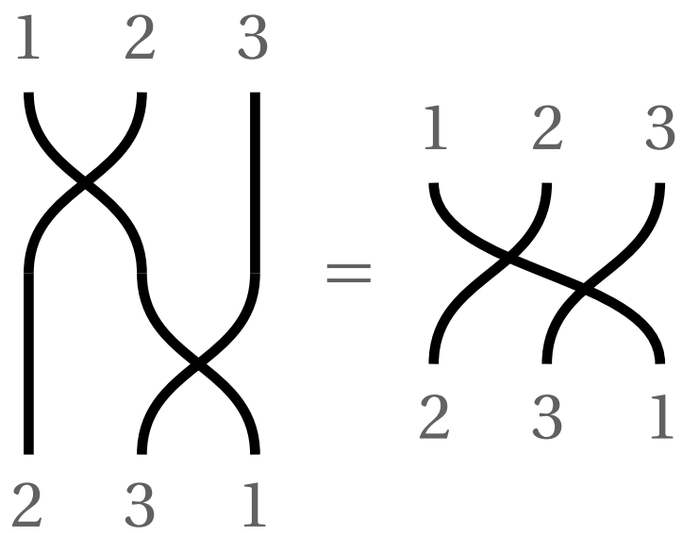
Scramblings and symmetries



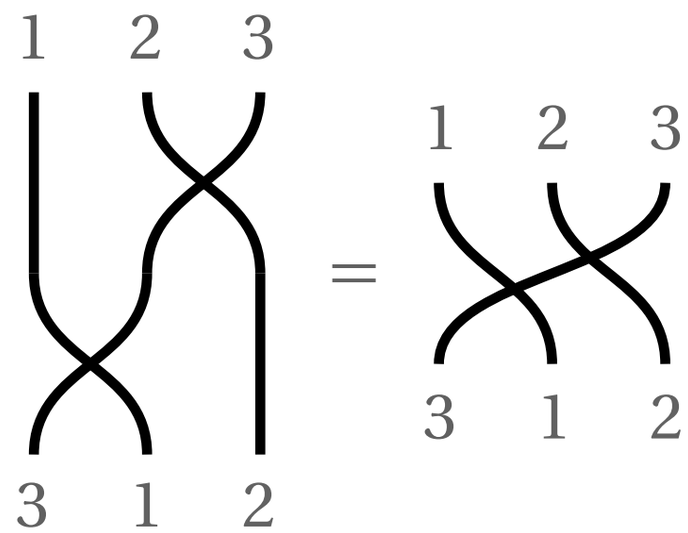


Scramblings of codes or networks are *permutations*

We can “multiply” them, but *xy* ≠ *yx*



but



Like a Rubik’s cube: order of rotations matters

From permutations to rotations

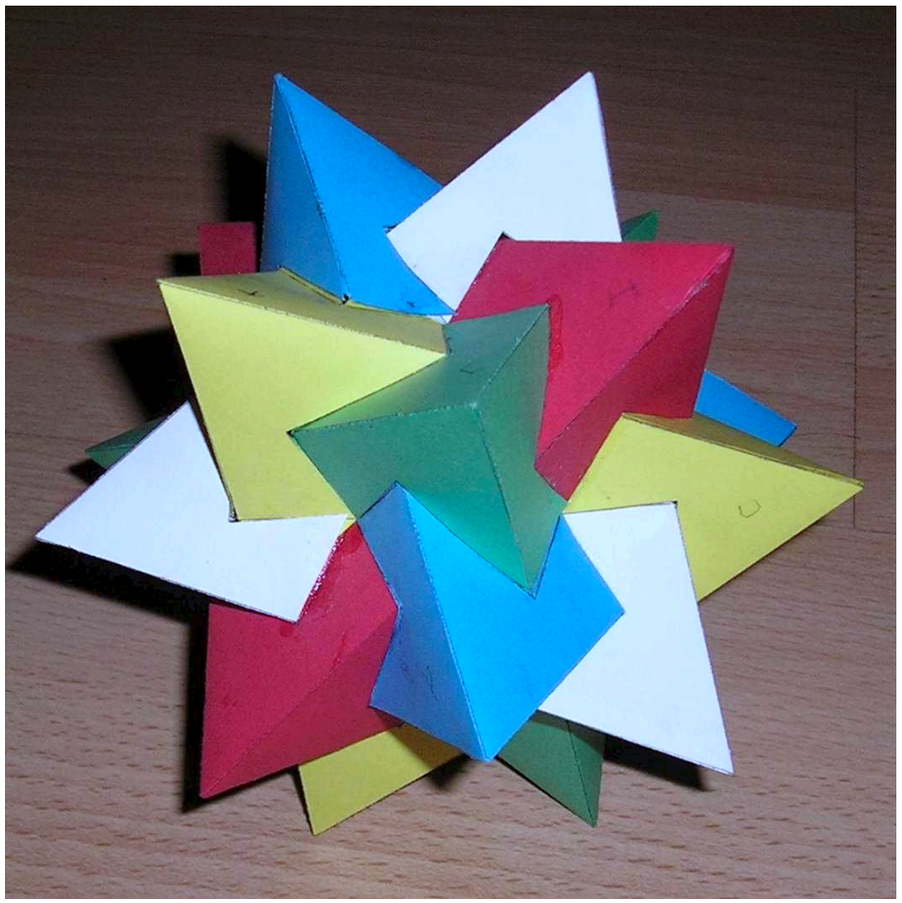
From permutations to rotations

Every group of permutations can be represented by rotations and reflections in a high-dimensional space

From permutations to rotations

Every group of permutations can be represented by rotations and reflections in a high-dimensional space

Permute the five colors:



The bad (or good, depending) news

The bad (or good, depending) news

The ways Alice can scramble her code correspond to rotations and reflections in a very (exponentially!) high-dimensional space

The bad (or good, depending) news

The ways Alice can scramble her code correspond to rotations and reflections in a very (exponentially!) high-dimensional space

If we try to use a method like Shor's, the algorithm "gets lost" in these spaces

The bad (or good, depending) news

The ways Alice can scramble her code correspond to rotations and reflections in a very (exponentially!) high-dimensional space

If we try to use a method like Shor's, the algorithm "gets lost" in these spaces

Any quantum measurement we try to do gives nearly the same result no matter what scrambling Alice used...

The bad (or good, depending) news

The ways Alice can scramble her code correspond to rotations and reflections in a very (exponentially!) high-dimensional space

If we try to use a method like Shor's, the algorithm "gets lost" in these spaces

Any quantum measurement we try to do gives nearly the same result no matter what scrambling Alice used...

...so we learn next to nothing about how to crack her code

What does this mean?

What does this mean?

To show that we can crack a code, one algorithm is enough

What does this mean?

To show that we can crack a code, one algorithm is enough

But to prove that we can’t, we have to reason about *all possible algorithms*

What does this mean?

To show that we can crack a code, one algorithm is enough

But to prove that we can't, we have to reason about *all possible algorithms*

Factoring, and breaking RSA, might be easy for classical computers:
maybe we just haven't been clever enough to think of an algorithm!
(We've been surprised before.)

What does this mean?

To show that we can crack a code, one algorithm is enough

But to prove that we can't, we have to reason about *all possible algorithms*

Factoring, and breaking RSA, might be easy for classical computers:
maybe we just haven't been clever enough to think of an algorithm!
(We've been surprised before.)

But any quantum algorithm that breaks the McEliece cryptosystem would have to use completely new ideas...

Alice, Bob, and Eve

Alice, Bob, and Eve

Alice, Bob, and Eve

If Eve measures Alice’s message, this disturbs it in a way that Alice and Bob can detect

Alice, Bob, and Eve

If Eve measures Alice’s message, this disturbs it in a way that Alice and Bob can detect

If their message got through without being intercepted, they can use it as a secret key [Bennett and Brassard, 1984]

Alice, Bob, and Eve

If Eve measures Alice’s message, this disturbs it in a way that Alice and Bob can detect

If their message got through without being intercepted, they can use it as a secret key [Bennett and Brassard, 1984]

144 km between the Canary Islands, and over 300 km of optical fiber: bank transfers and election results in Geneva

So, can we build them?

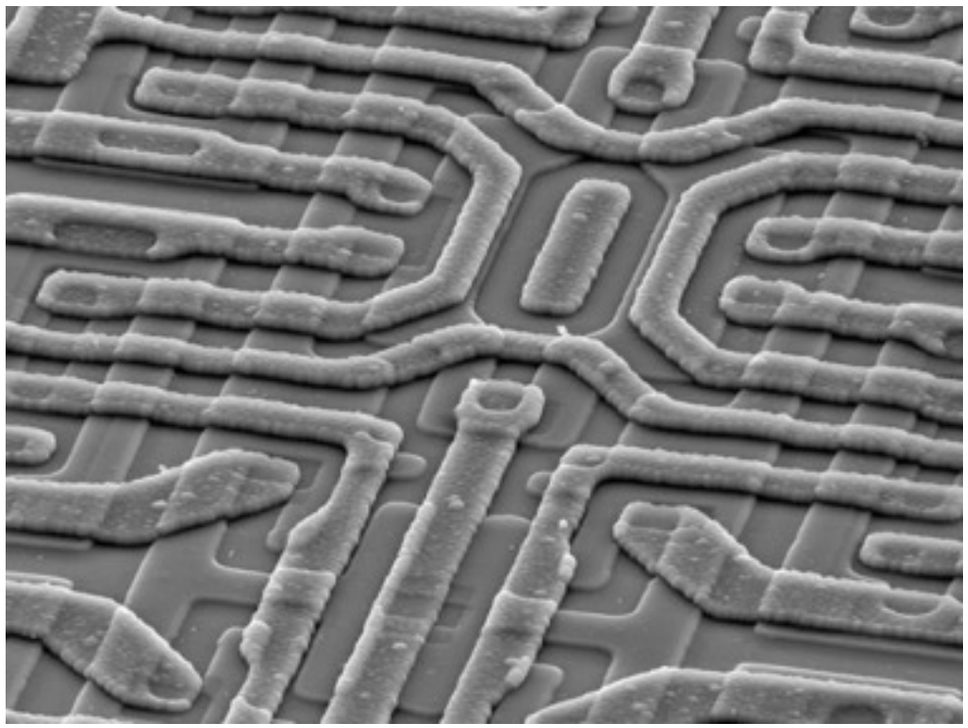
So, can we build them?

The first transistor looked like this:

So, can we build them?

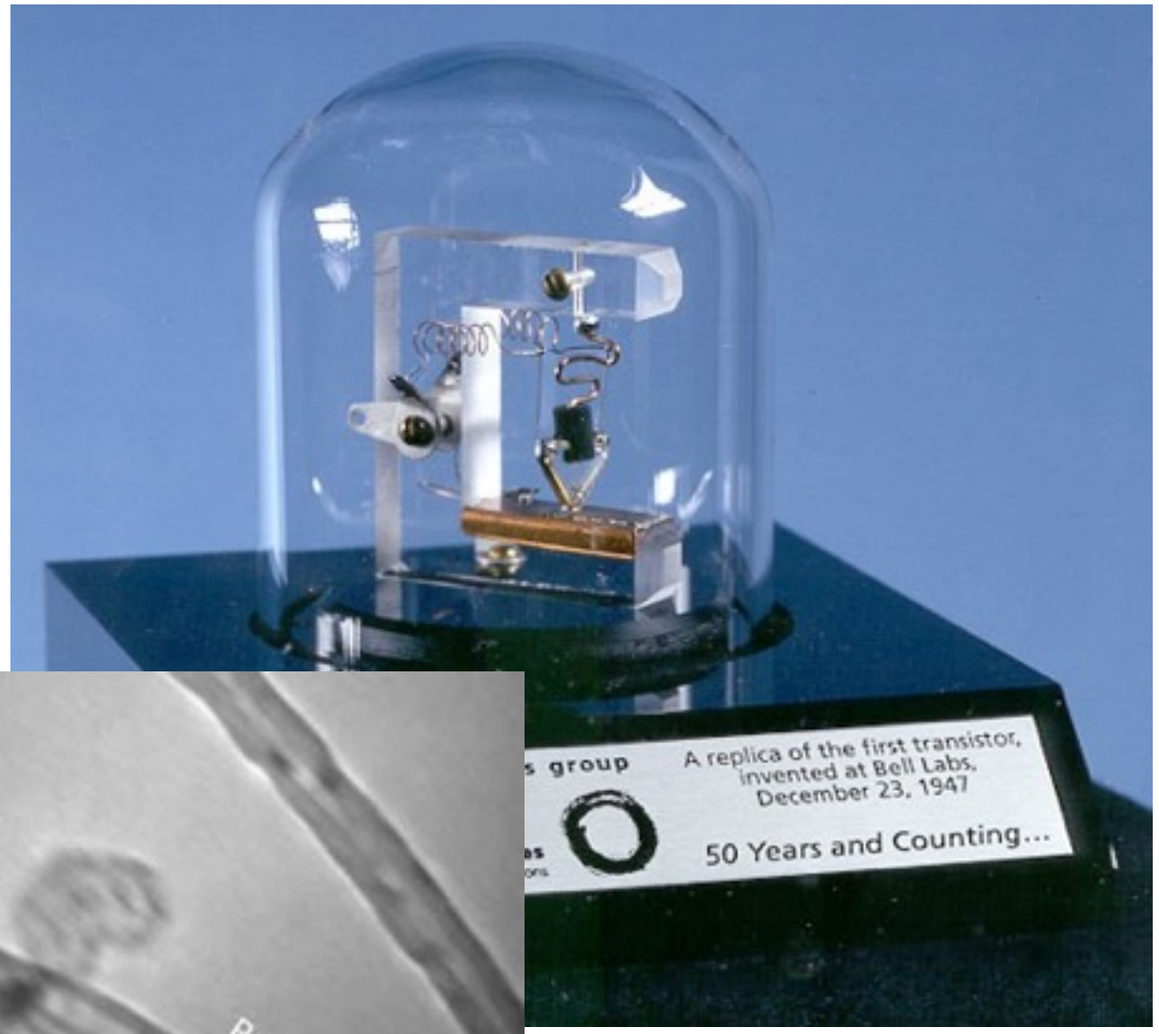
The first transistor looked like this:

Now they look like this:

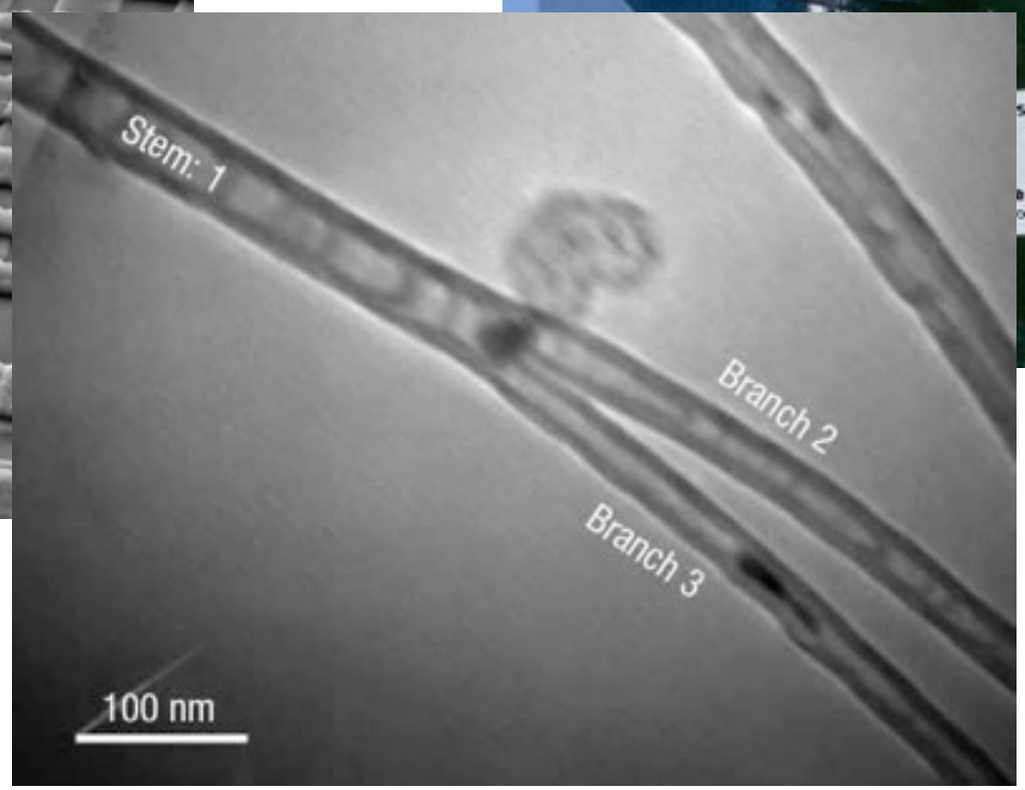


So, can we build them?

The first transistor looked like this:



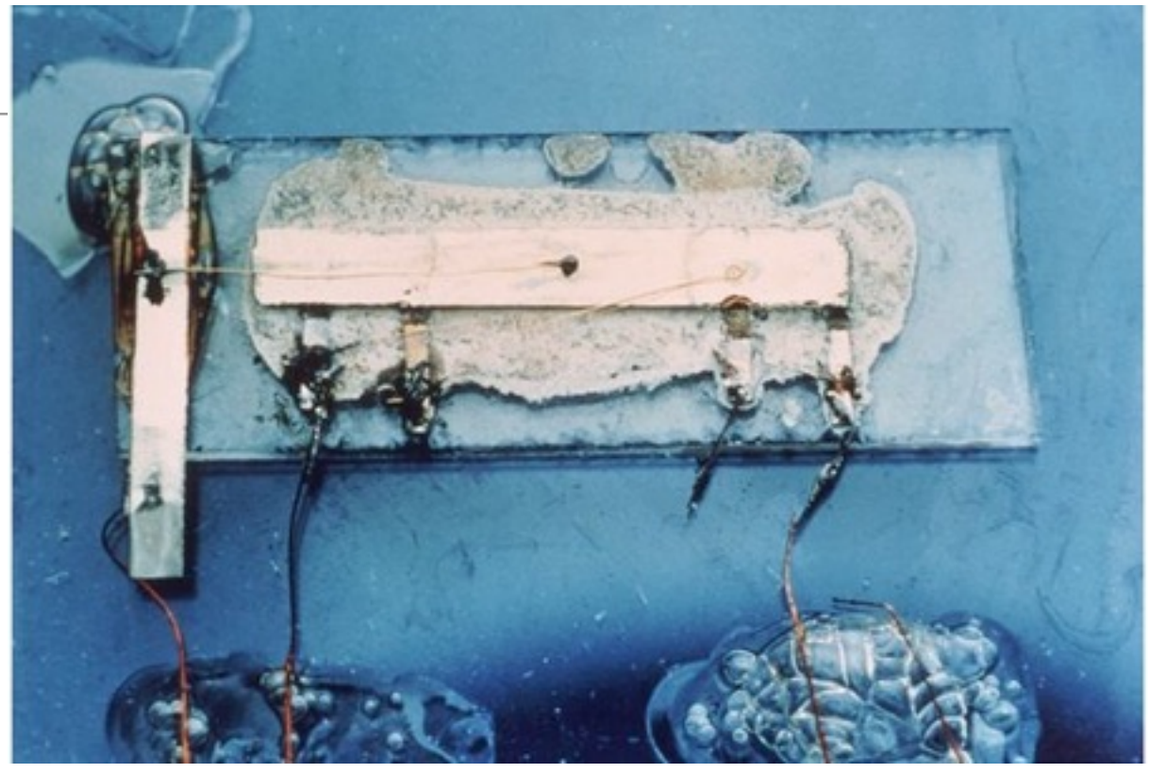
Now they look like this:



So, can we build them?

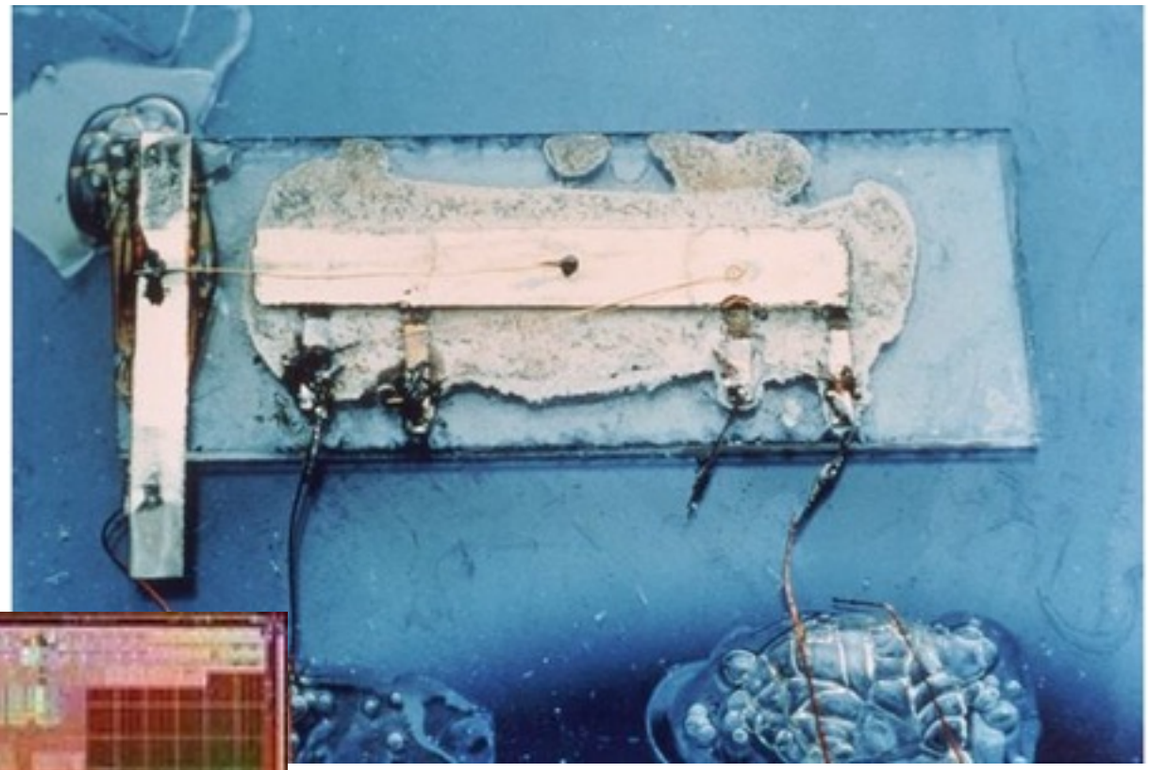
So, can we build them?

The first integrated circuit looked like this:

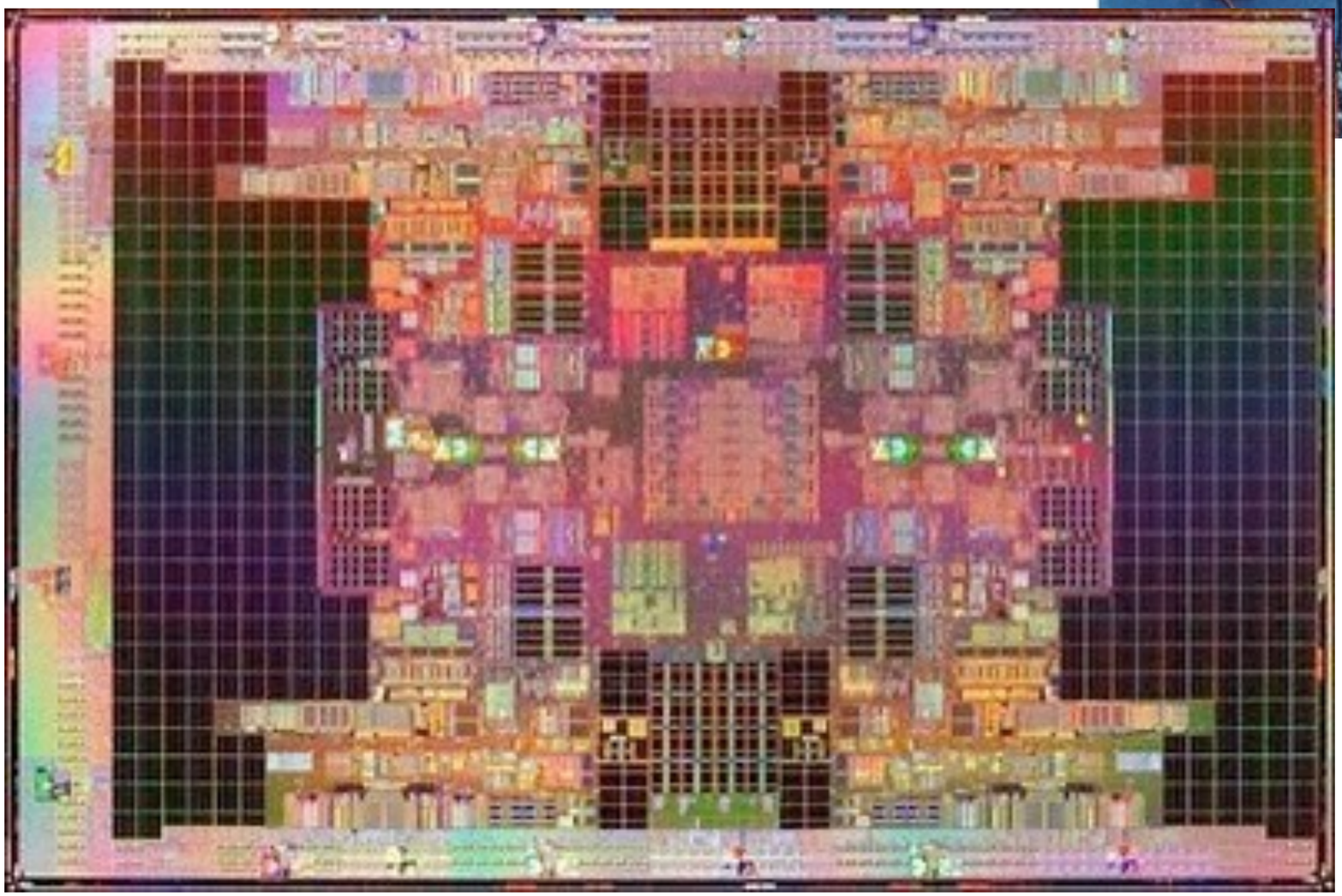


So, can we build them?

The first integrated circuit looked like this:



Now they look like this:



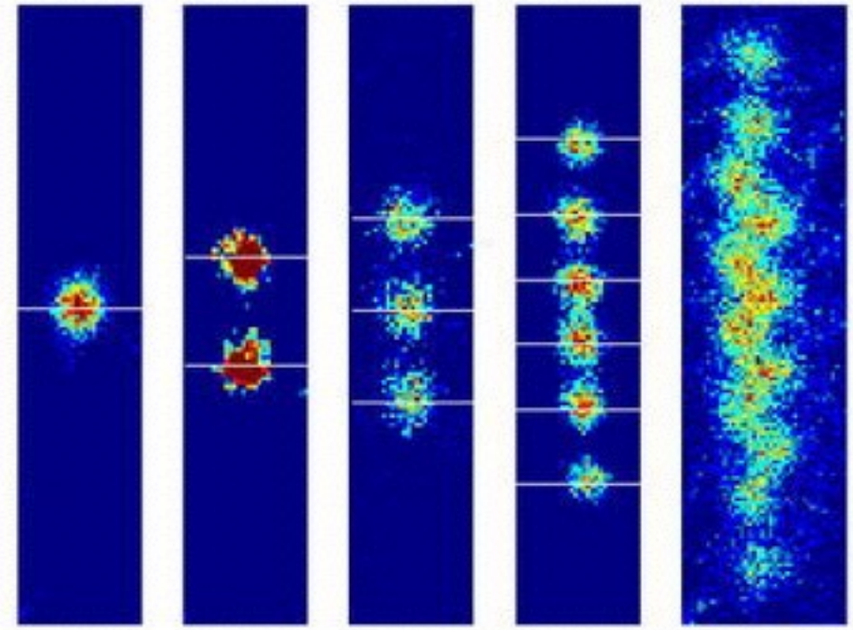
So, can we build them?

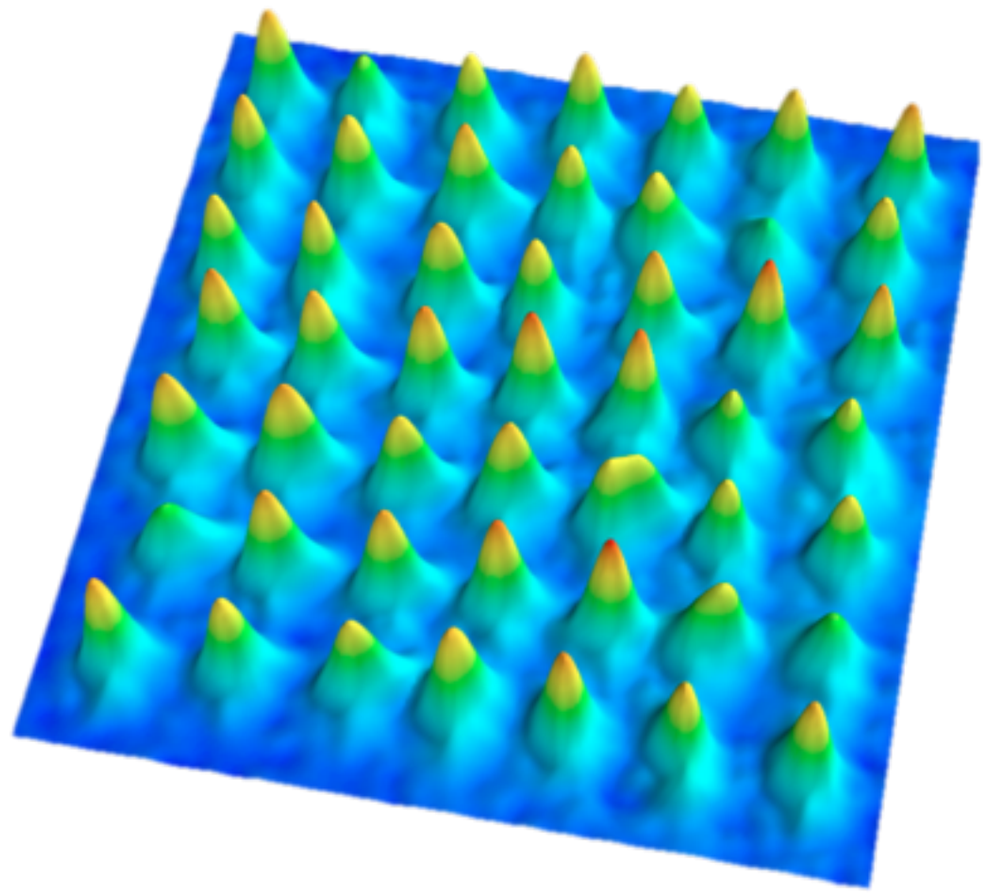
So, can we build them?

We are rapidly gaining an ability to manipulate single atoms

So, can we build them?

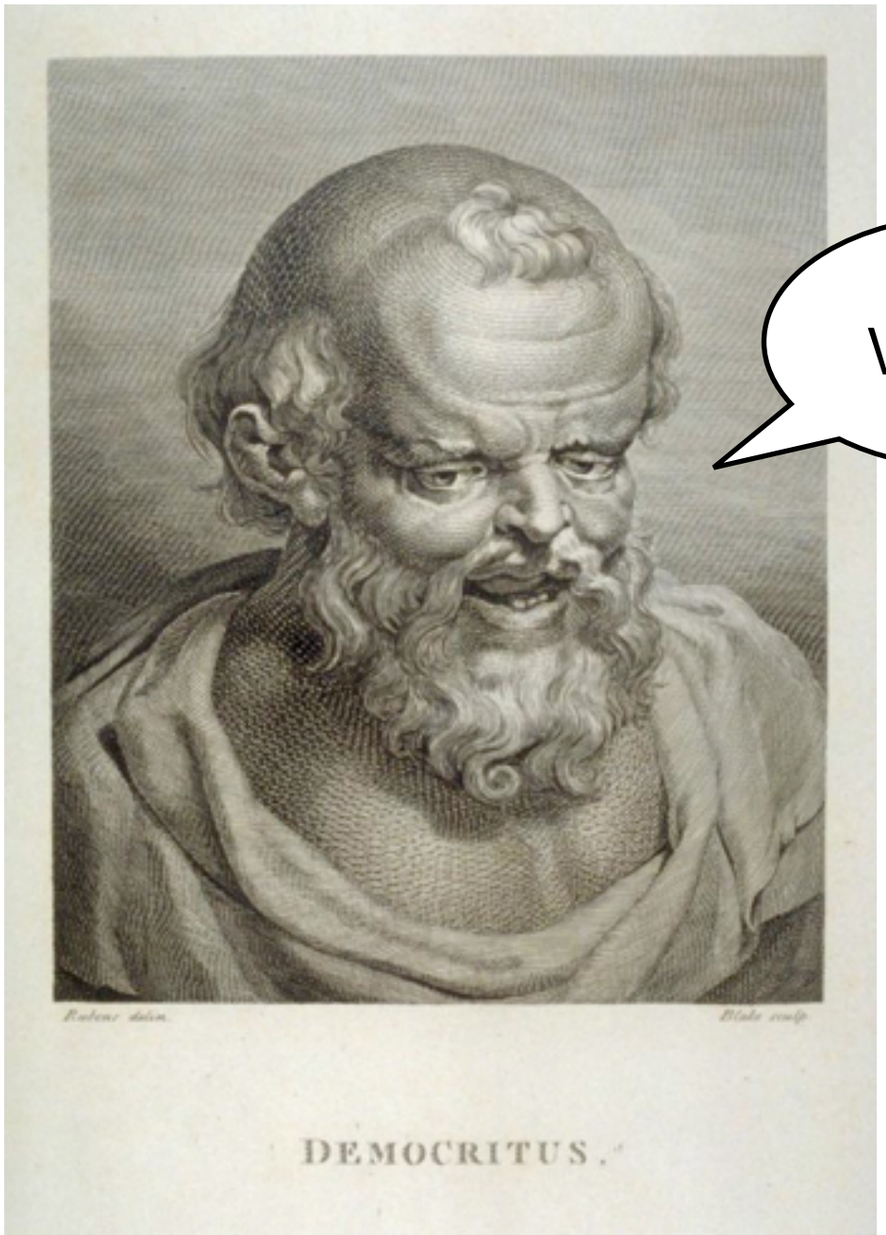
We are rapidly gaining an ability to manipulate single atoms

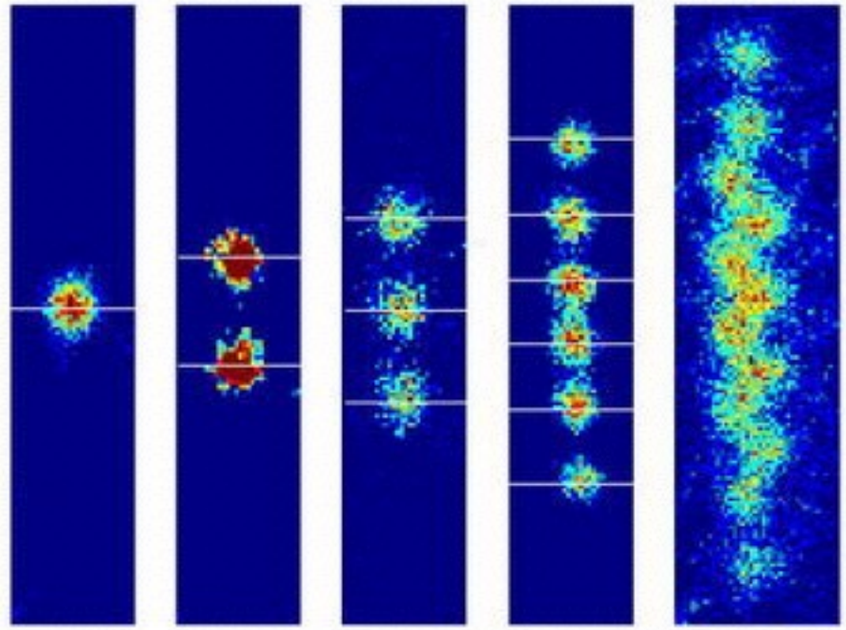


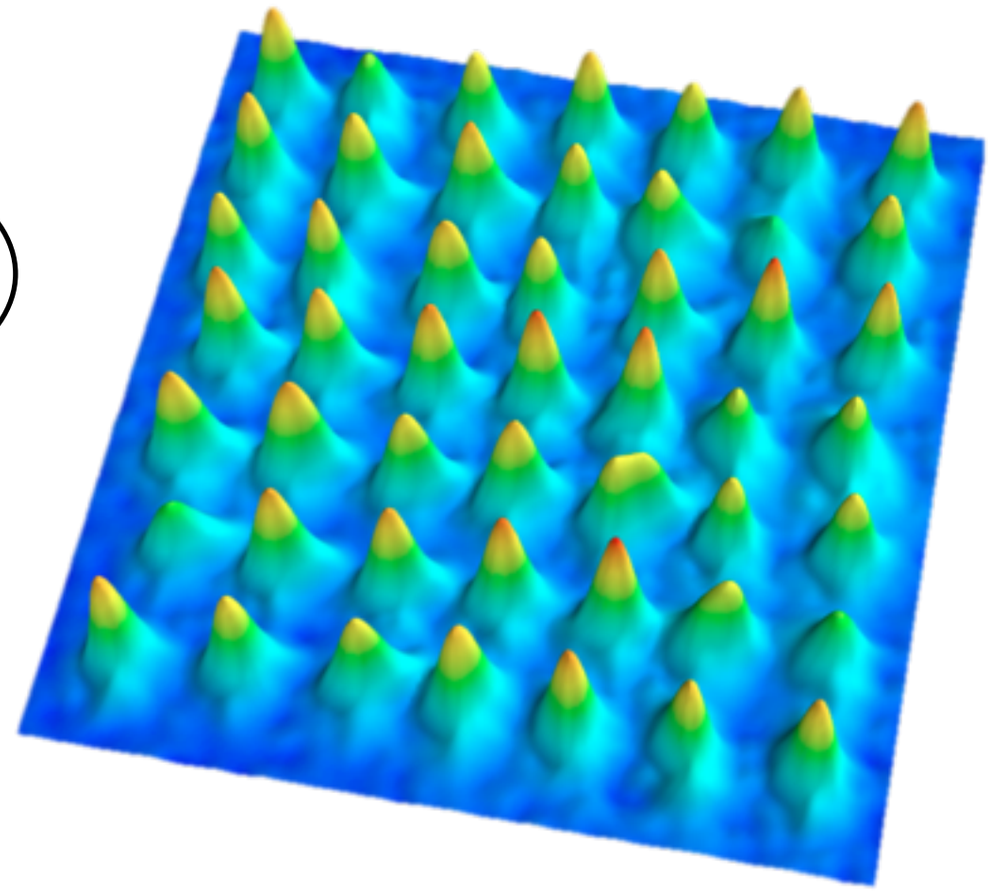


So, can we build them?

We are rapidly gaining an ability to manipulate single atoms

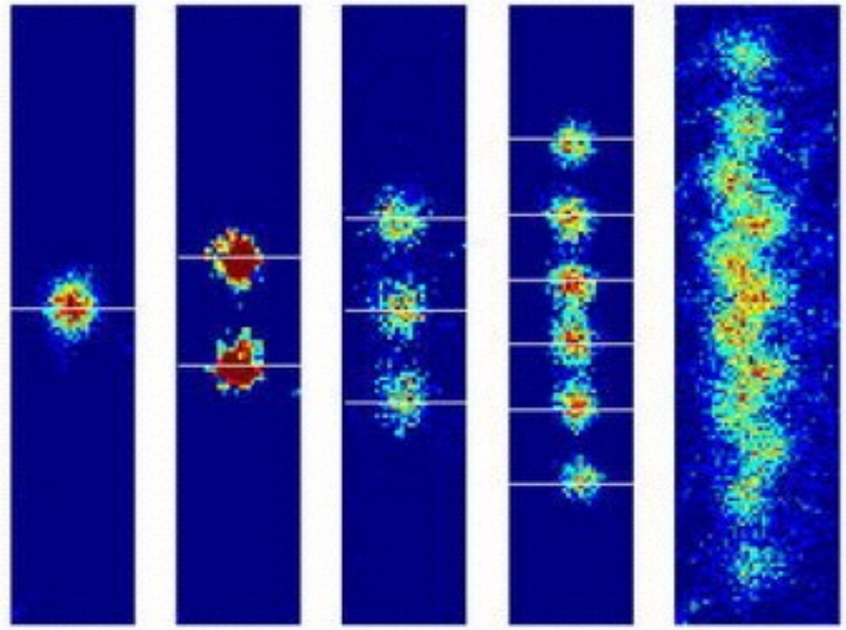




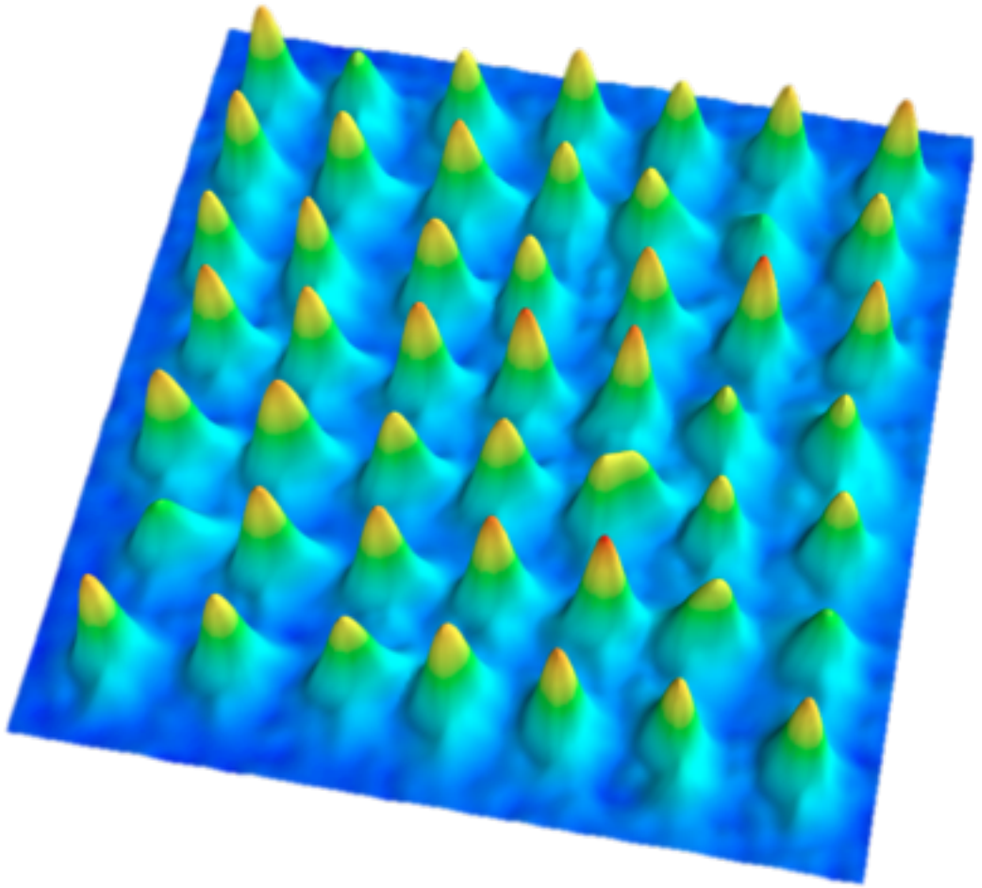


So, can we build them?

We are rapidly gaining an ability to manipulate single atoms

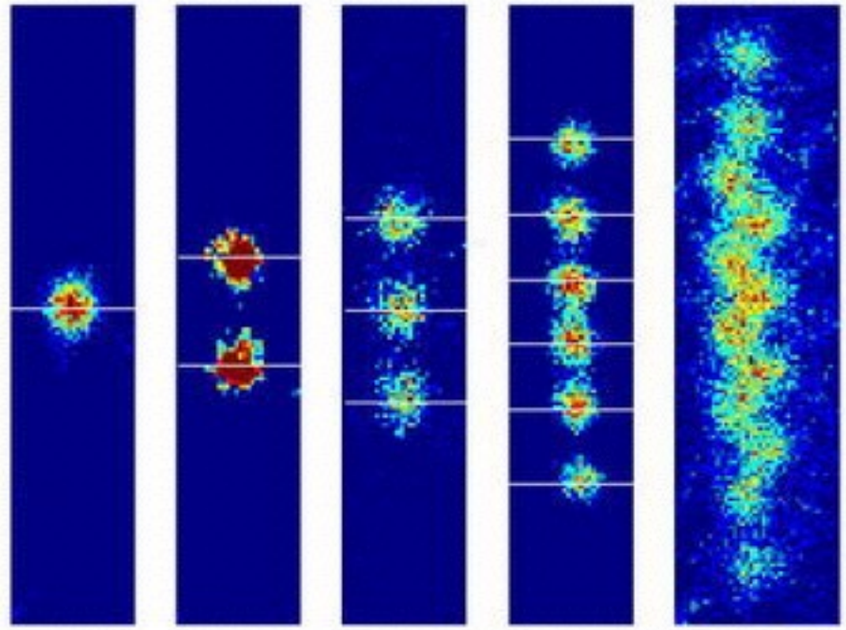


Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state



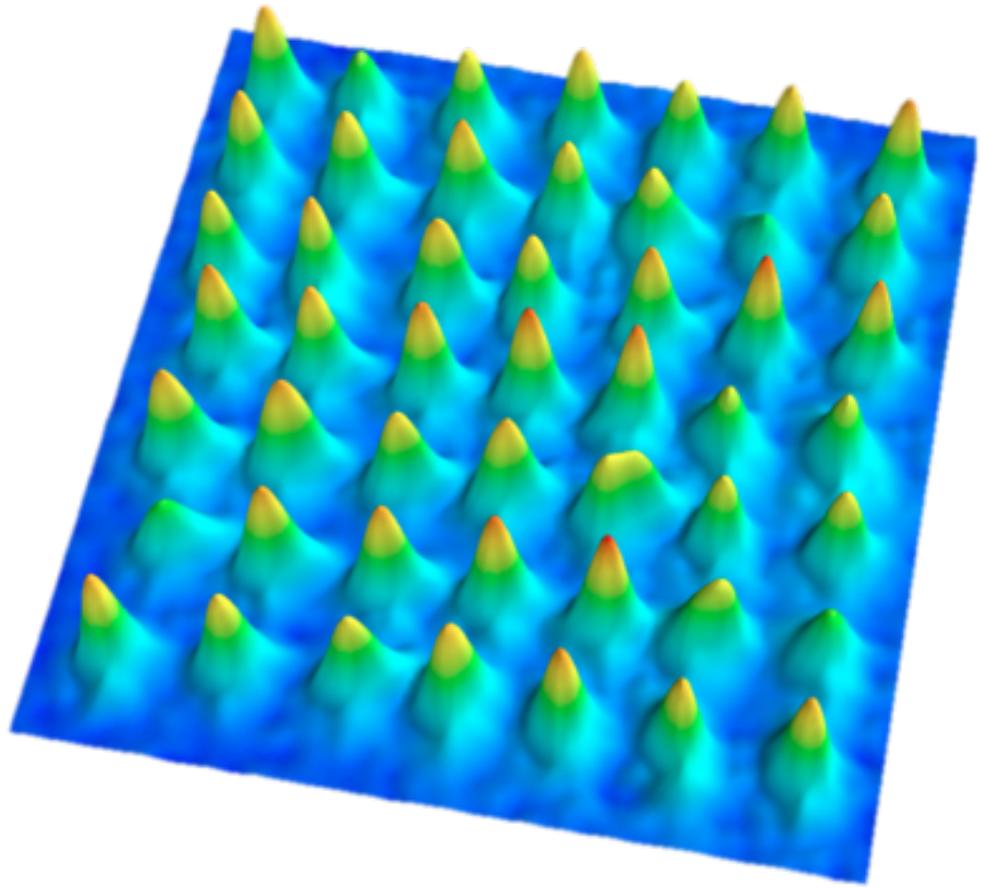
So, can we build them?

We are rapidly gaining an ability to manipulate single atoms

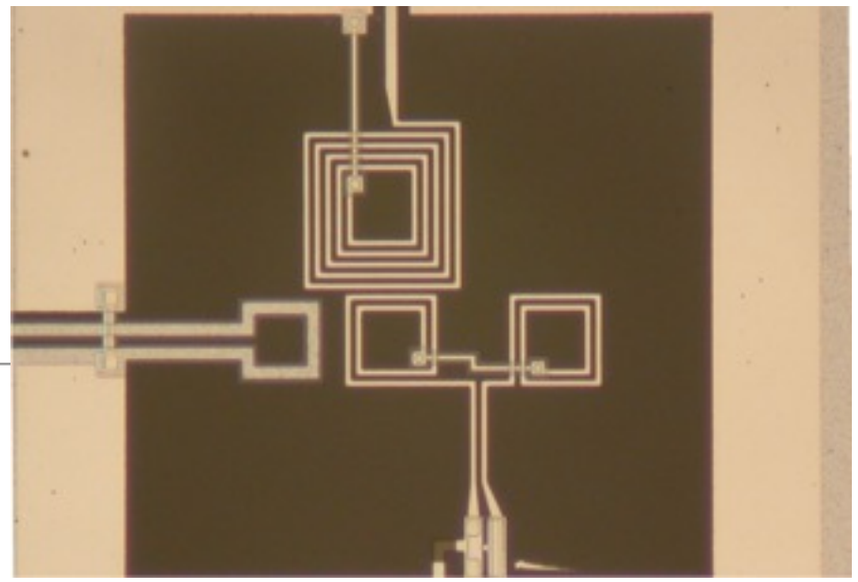


Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state

This is challenging, but not impossible: simple quantum devices are now being built



So, can we build them?



We are rapidly gaining an ability to manipulate single atoms

Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state

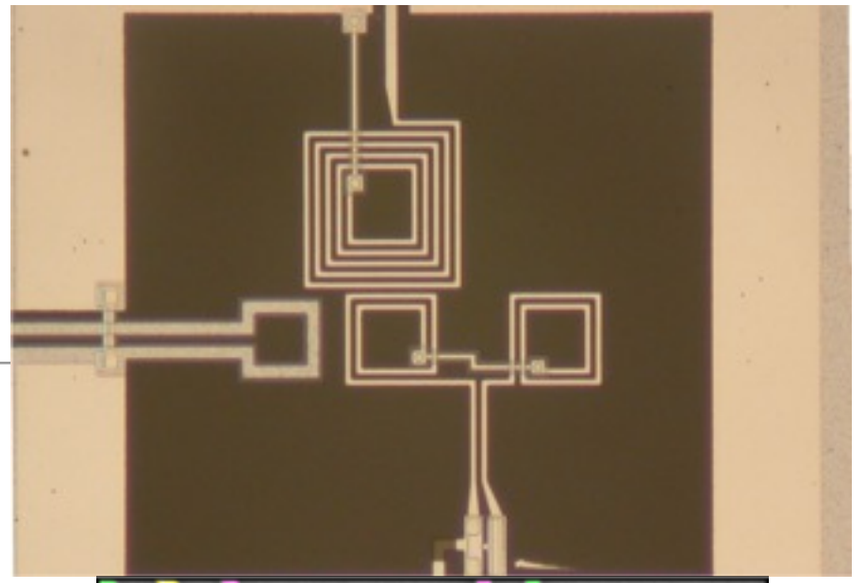
This is challenging, but not impossible:
simple quantum devices are now being built

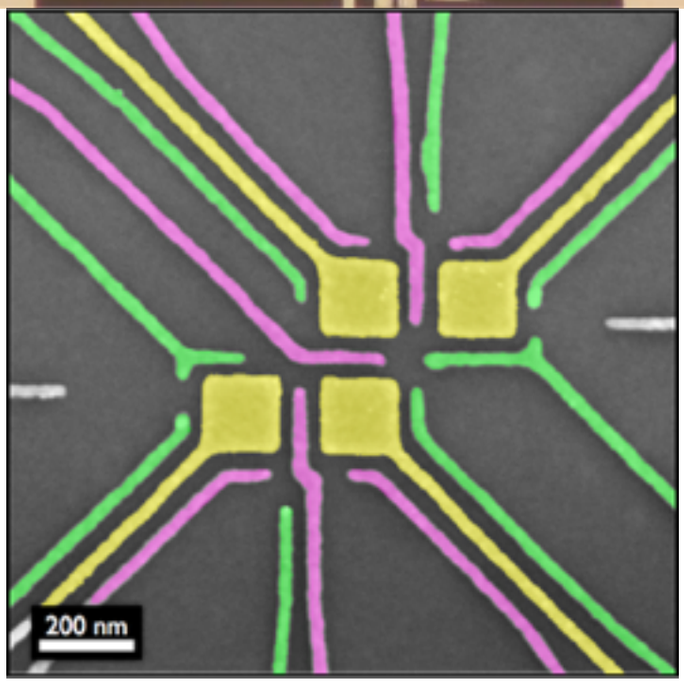
So, can we build them?

We are rapidly gaining an ability to manipulate single atoms

Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state

This is challenging, but not impossible: simple quantum devices are now being built



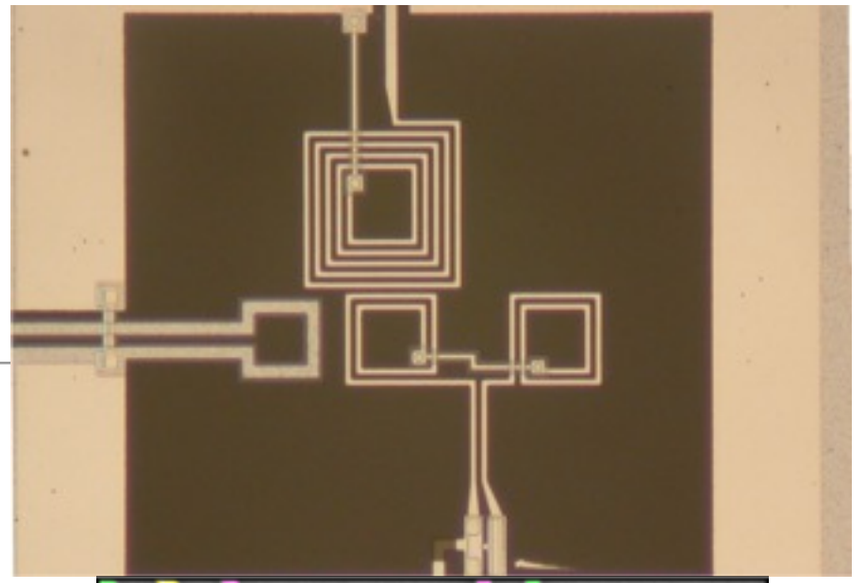


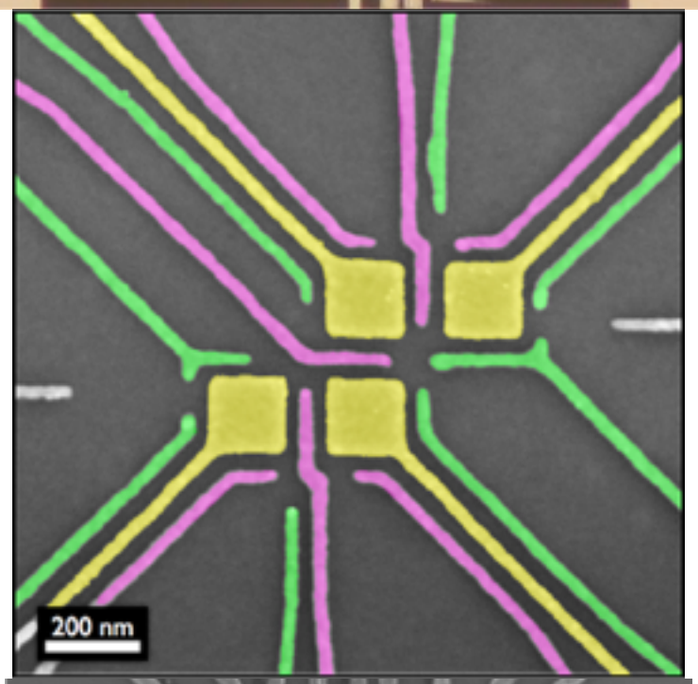
So, can we build them?

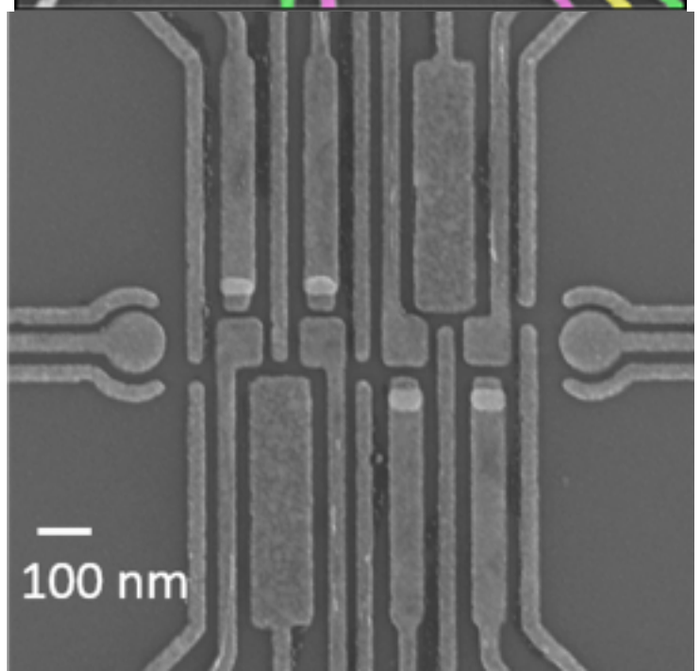
We are rapidly gaining an ability to manipulate single atoms

Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state

This is challenging, but not impossible: simple quantum devices are now being built







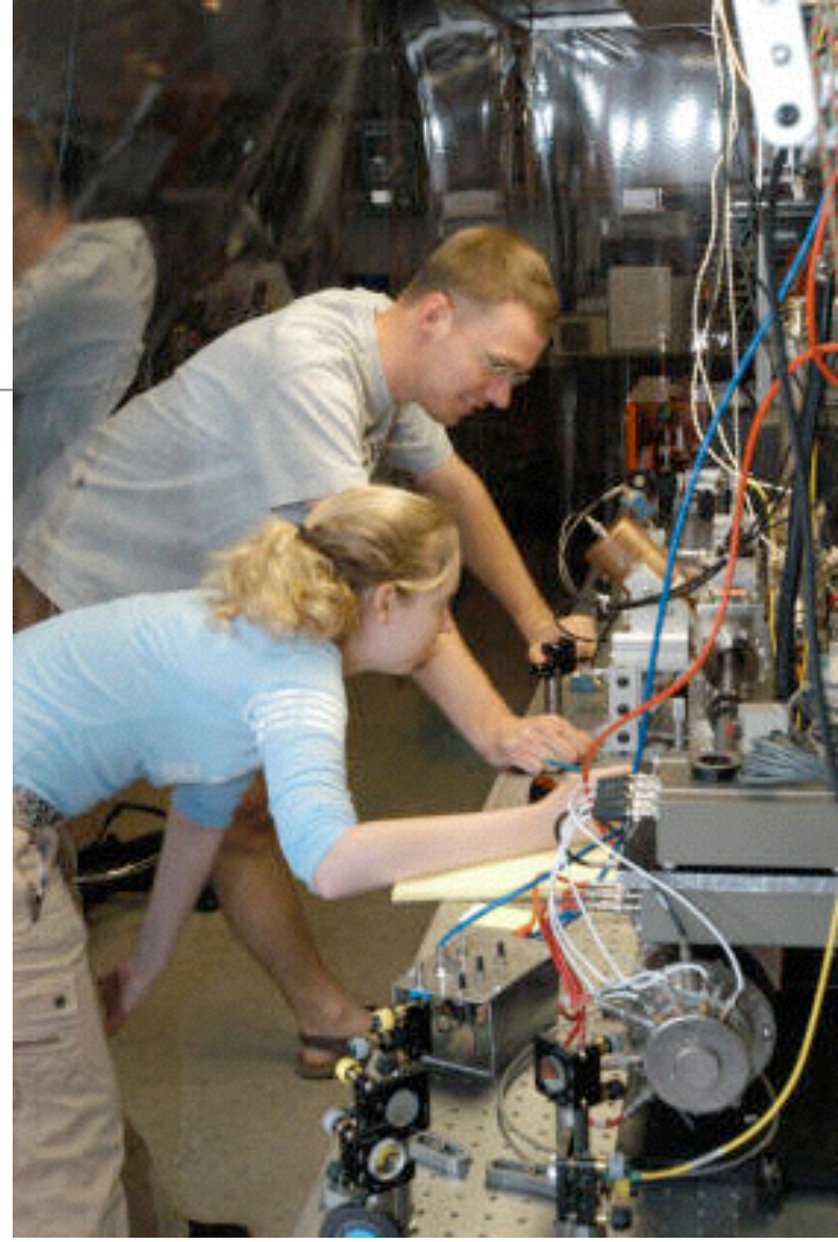
So, can we build them?

We are rapidly gaining an ability to manipulate single atoms

Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state

This is challenging, but not impossible: simple quantum devices are now being built

It will take a while for this...



So, can we build them?

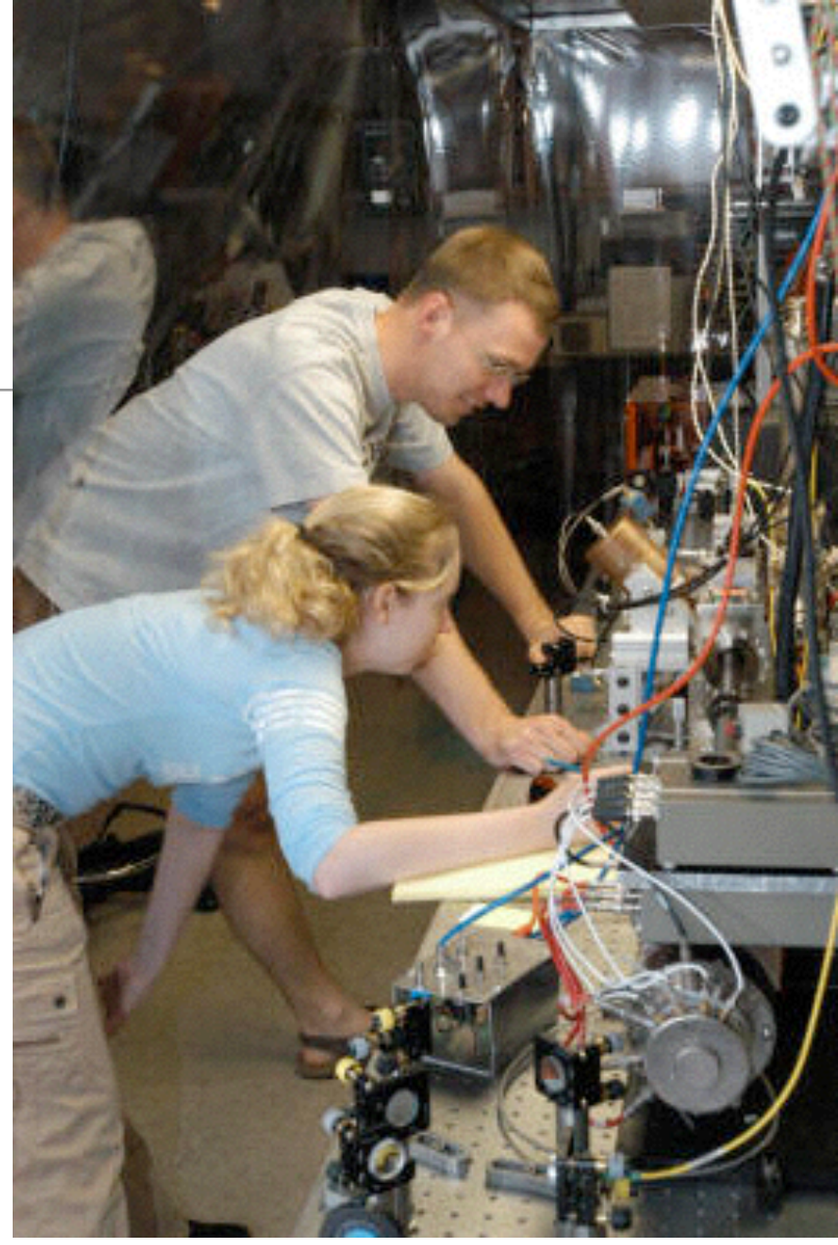
We are rapidly gaining an ability to manipulate single atoms

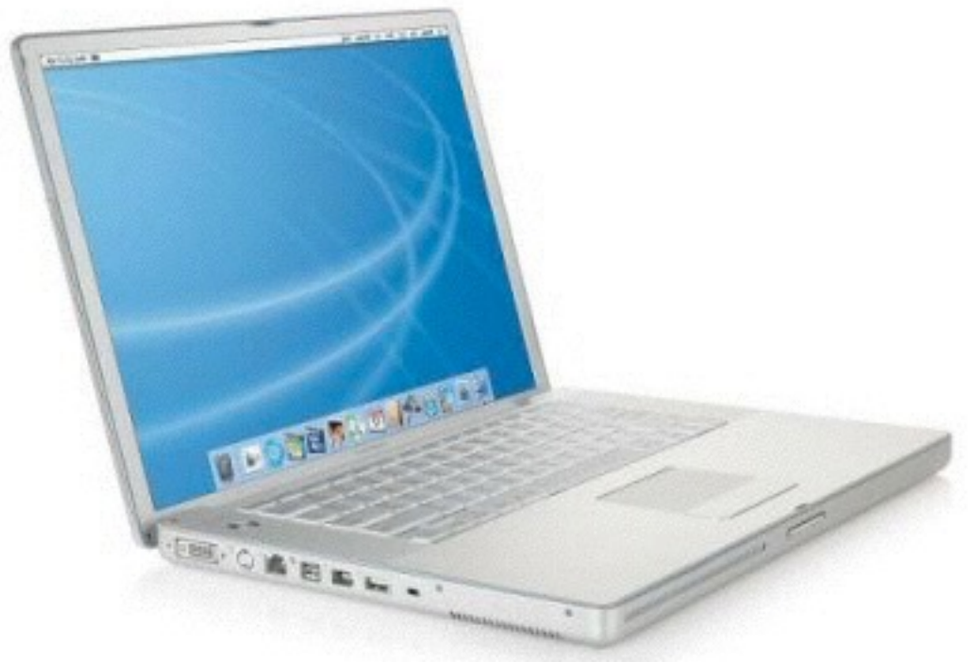
Quantum computation is delicate: we need strong interactions in order to compute, but we have to avoid stray interactions that “measure” and destroy the quantum state

This is challenging, but not impossible: simple quantum devices are now being built

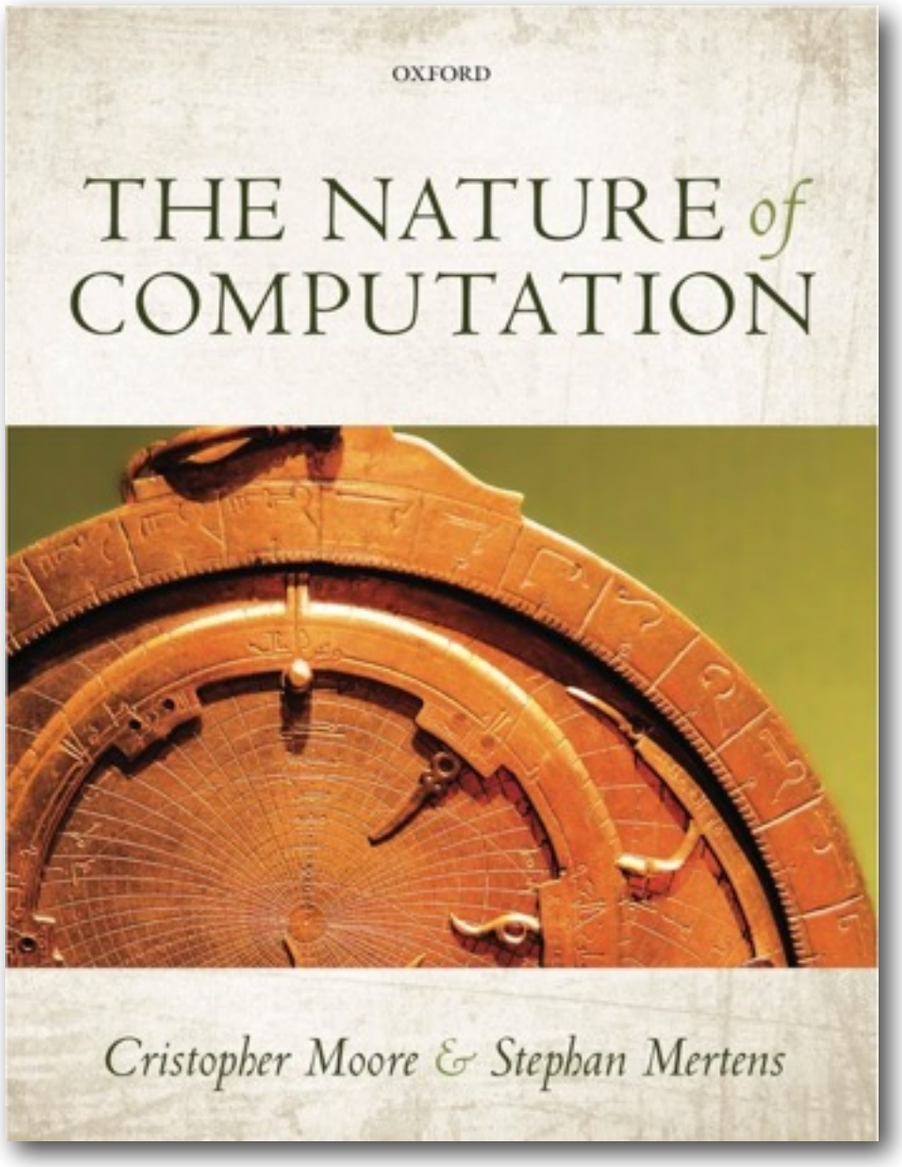
It will take a while for this...

To turn into this





Shameless Plug



www.nature-of-computation.org

To put it bluntly: this book rocks! It somehow manages to combine the fun of a popular book with the intellectual heft of a textbook.

Scott Aaronson, MIT

This is, simply put, the best-written book on the theory of computation I have ever read; one of the best-written mathematical books I have ever read, period.

Cosma Shalizi, Carnegie Mellon