Mathematics for Computer Science

Prof.dr.hab. Viorel Bostan

Technical University of Moldova

viorel.bostan@adm.utm.md

Lectures 7–8

Cryptography: First encounter

- What was the first encounter in your life with cryptography issues?
- e.g. some sort of communicating a secure protocol, through some insecure communication channel, its breach and its catastrophic consequences?
- Hint: probably it happened at the age of 2–3.

– Dragii mamei copilași! Eu mă duc în pădure ca să mai duc ceva de-a mâncării. Dar voi, încuieți ușa după mine, ascultați unul de altul, și să nu cumva să deschideți până ce nu-ți auzi glasul meu. Când voi veni eu, am să vă dau de știre, ca să mă cunoașteți, și am să vă spun așa:

Trei iezi cucuieți
Ușa mamei descuieți!
Că mama v-aduce vouă:
. . . . . .

Auzit-ați ce-am spus eu?
– Da, mămucă, ziseră iezii.
– Pot să am nădejde în voi?
– Să n-ai nici o grijă, mămucă, apucară cu gura î nainte cei mai mari. Noi suntem odată băieți, și ce-am vorbit odată vorbit rămâne.

Picture of the day

Definition

Cryptography is about communication in the presence of an adversary.

It encompasses many problems like encryption, authentication, key distribution and others:

- Theoretical foundation (math):
  - Number Theory;
  - Probability Theory;
  - Computational Complexity Theory;
  - Information Theory;
- Sets the problem;
- Defines secure protocols;
- Defines ways to build secure protocols;
- Ensure confidence in security.

Encryption: historical glance

- Most ancient and basic problem of cryptography is

  Secure communication over an insecure channel.
- Party **A** wants to send a secret message to party **B** over a communication line which may be tapped by an adversary.
- Traditional solution to this problem is called the **private key encryption**.
- In private key encryption **A** and **B** hold a meeting before the remote transmission takes place and agree on a pair of encryption and decryption algorithms (or functions) E and D, and an additional piece of information S to be kept secret, called **common secret key**.
- The adversary may know the algorithms E and D, but adversary doesn't know secret key S. So he can not read the message.

Encryption: historical glance

- After the initial meeting when **A** wants to send **B** the *clear-text* or *plain-text* message m over the insecure communication line, **A** encrypts m by computing the *cipher-text*

$$c = E(S, m)$$

  and sends c to **B**.
- Upon receipt, **B** decrypts c by computing

$$m = D(S, c).$$

- The adversary, who does not know the secret key S, should not be able to compute message m from c.

Encryption: historical glance

- One of the oldest encryption algorithms is **the substitution cipher.**
- In substitution cipher **A** and **B** meet and agree on some secret permutation

$$f : \Sigma \to \Sigma,$$

  where Σ is the alphabet of the message being sent.
- To encrypt message $m_1 m_2 \dots m_n$, where $m_i \in \Sigma$, party **A** computes

$$E(f, m) = f(m_1) f(m_2) \dots f(m_n) = c_1 c_2 \dots c_n \equiv c.$$

- To decrypt message $c_1 c_2 \dots c_n$, where $c_i \in \Sigma$, party **B** computes

$$D(f, m) = f^{-1}(c_1) f^{-1}(c_2) \dots f^{-1}(c_n) = m_1 m_2 \dots m_n \equiv m.$$

- The common secret key here is the permutation f.

- Example of such permutation is

$$\begin{array}{cccccccccccc} A & B & C & D & E & F & G & I & J & K & L\ldots \\ W & I & F & G & B & A & Q & N & C & D & U\ldots \end{array}$$

- The secret key S is the permutation and can be given by a table.
- To decrypt the message, the inverse permutation is used

$$\begin{array}{cccccccccccc} A & B & C & D & E & F & G & I & J & K & L\ldots \\ F & E & J & K & M & C & D & B & S & V & P\ldots \end{array}$$

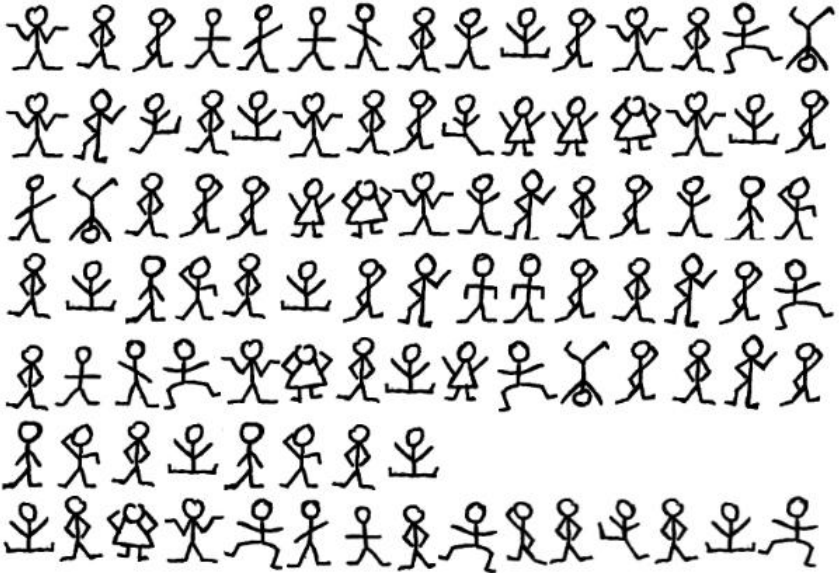
- Is it easy to guess the permutation? If 26 letters are being used, then the number of all permutations is 26!
- To simplify the key, the **translation permutation (code)** can be used:

$$\begin{array}{cccccccccccc} A & B & C & D & E & F & G & I & J & K & L\ldots \\ D & E & F & G & I & J & K & L & M & N & O\ldots \end{array}$$

- Here permutation is obtained by transposing the letters with step 3, which is the secret key S. In other words, $S = 3$.

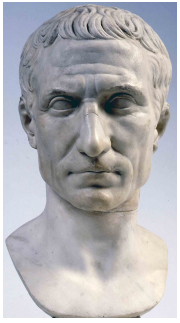
Encryption: historical glance

- The translation (transposition) code is easy to break: all you have to do is to check for all possible steps: $1, 2, 3, 4, \ldots, 25$.
- In the general case, the substitution cipher is easy to break by an adversary who sees a moderate (as a function of the size of the alphabet Σ) number of cipher-texts and knows the language being used.
- If you recall "The Adventure of the Dancing Men", one of the 56 Sherlock Holmes short stories written by British author Sir Arthur Conan Doyle, you will see a modification of substitution cipher, where all letters of English alphabet where substituted with pictographs depicting dancing figures.
- Once you know the language in which the original message was written, and you have at hand several messages, you can break the code. Ask Sherlock!



Encryption: historical glance

According to Suetonius, Julius Caesar, used a simple substitution cipher with a shift of 3 or 4.



"If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others."

Suetonius, Life of Julius Caesar, 56

Encryption: historical glance

Actually, Caesar Code can be traced back to Antique Greece.
Skytale is a coding device known before Caesar was born.

The secret key is the diameter of cylinder.

Example

For example: the plain-text is:

HELP ME I AM UNDER ATTACK

Suppose the rod allows one to write 4 letters around in a circle and 5 letters along the rod.

H	E	L	P	M
E	I	A	M	U
N	D	E	R	A
T	T	A	C	K

Then the message sent will be:

HENTEIDTLAEAPMRCMUAK

Does Perfect Secrecy Exist?

A rigorous theory of perfect secrecy based on information theory was developed by *Claude E. Shannon* in 1943.

In this theory, the adversary is assumed to have unlimited computational resources.

Shannon showed that secure (properly defined) encryption system can exist only if the size of the secret key S that **A** and **B** agree on prior to remote transmission is as large as the number of secret bits to be ever exchanged remotely using the encryption system.

An example of a private key encryption method which is secure even in presence of a computationally unbound adversary is the **one time pad** method.

Encryption: one time pad

A and **B** agree on a secret bit string

$$pad = b_1 b_2 \dots b_n,$$

where $b_i \in \{0, 1\}$ (i.e pad is chosen in $\{0, 1\}^n$ with uniform probability).
This is the common secret key. To encrypt a message $m_1 m_2 \dots m_n$, where $m_i \in \{0, 1\}$, party **A** computes

$$E(pad, m) = m \oplus pad = c,$$

where $\oplus$ denotes the bitwise **exclusive or**.
To decrypt cipher-text $c \in \{0, 1\}^n$, party **B** computes

$$D(pad, c) = pad \oplus c = pad \oplus (m \oplus pad) = m.$$

It can be shown that if cipher-text c is known, then the probability of guessing message m equals $\frac{1}{2^n}$.
Seeing cipher-text c gives "no information" about what has been sent, i.e. that the adversary's aposteriori probability of predicting m given c is no better than his apriori probability of predicting m without being given c.

Encryption: one time pad

Suppose **A** wants to send **B** an additional message m'.

If **A** were to simply send $c = E(pad, m')$, then the sum of the lengths of messages m and m' will exceed the length of the secret key pad, and thus by Shannon's theory the system cannot be secure.

The adversary can compute $E(pad, m) \oplus E(pad, m') = m \oplus m'$, which gives information about m and m'.

For example, one can tell which bits of m and m' are equal and which are different.

To fix this, the length of the pad agreed upon apriori should be the sum total of the length of all messages ever to be exchanged over the insecure communication line.

Minimal Requirement of Security

What is the meaning of the terms "secure" and "break the system"?

It is clear that a minimal requirement of security would be that:

> *any adversary who can see the cipher-text and knows which encryption and decryption algorithms are being used, can not recover the entire clear-text.*

But, many more properties may be desirable.

1. It should be hard to recover the messages from the cipher-text when the messages are drawn from arbitrary probability distributions defined on the set of all strings.
   We must assume that the message space is known to the adversary.
2. It should be hard to compute partial information about messages from the cipher-text.
3. It should be hard to detect simple but useful facts about traffic of messages, such as when the same message is sent twice.
4. The above properties should hold with high probability.

Foundation of cryptography

Backbone of modern cryptography is the **Number Theory**.

Basic premise:

Multiplying two numbers is a simple process, but factoring the product back into the original two numbers is much more difficult to do computationally. Difficulty increases as we use larger numbers.

Definition

Number theory is area of mathematics that studies integers.

Assumption

For the next 2 lectures, we will consider only integer numbers.

Theorem (Division Theorem)

Given two integers a and b, with $a \leq b$, there exist unique integers q and r such that

$$b = a \cdot q + r, \quad \text{with } 0 \leqslant r < a.$$

Integer q is called **quotient** *and integer r is called* **reminder**.

Example

If $b = 19$ and $a = 5$, then $19 = 3 \cdot 5 + 4$.
If $b = 100$ and $a = 11$, then $100 = 9 \cdot 11 + 1$.

Definition

If remainder is $r = 0$, we are saying that b is divisible by a,

denoted by $b \vdots a$, and a is a divisor of b denoted by $a \mid b$.

Definition

a divides b iff $ak = b$ for some k.

Synonymous phrases

The following phrases are saying the same thing:

- a divides b,
- a is a divisor of b,
- a is a factor of b,
- b is divisible by a,
- b is a multiple of a.

$$\forall n \quad n \mid 0, \quad n \mid n, \quad \pm 1 \mid n, \quad 0 \mid n \text{ Implies } n = 0.$$

Definition

A number is said to be **perfect** if it equals the sum of its positive divisors, excluding itself.

Example

Perfect numbers: $6 = 1 + 2 + 3$ and $28 = 1 + 2 + 4 + 7 + 14$,
Nor perfect: $8 \neq 1 + 2 + 4$ and $12 \neq 1 + 2 + 3 + 4 + 6$.

Open question

Is there an odd perfect number?

All numbers up to about 10^{300} have been ruled out, but no one has proved that there isn't an odd perfect number waiting to be found. Or, nobody proved that there is an odd perfect number.

Lemma

Let a, b and c be integers. Then,

1. *If $a \mid b$ and $b \mid c$, then $a \mid c$.*
2. *If $a \mid b$ and $a \mid c$, then $a = sb + tc$ for all s and t.*
3. *For all $c \neq 0$, $a \mid b$ if and only if $ca \mid cb$.*

Proof of Property 2.

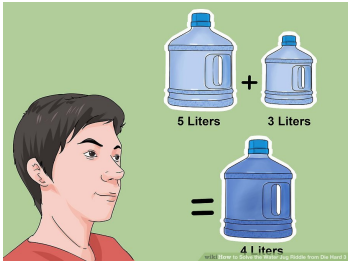
Given that $a \mid b$, there is some k_1 such that $ak_1 = b$.
Similarly, $a \mid c$, implies that there is some k_2 such that $ak_2 = c$.
Therefore,

$$sb + tc = s(ak_1) + t(ak_2) = (sk_1 + tk_2)a$$

Thus, $sb + tc = k_3 a$, where, $k_3 = sk_1 + tk_2$, which means that $a \mid sb + tc$.

Expression $sb + tc$ is called **linear combination** of b and c. □

Water Jugs Problem



Water Jugs Problem

$(0,0) \rightarrow (3,0)$	fill 3L jug
$\rightarrow (0,3)$	pour 3L jug into 5L jug
$\rightarrow (3,3)$	fill 3L jug
$\rightarrow (1,5)$	pour 3L jug into 5L jug
$\rightarrow (1,0)$	empty 5L jug
$\rightarrow (0,1)$	pour 3L jug into 5L jug
$\rightarrow (3,1)$	fill 3L jug
$\rightarrow (0,4)$	pour 3L jug into 5L jug

General Water Jugs Problem

Suppose we have one jug of a Liters and a second jug of b Liters. Suppose that $b \geqslant a$.
Is it possible to measure c Liters?

For example, having 6 and 10 Liters water jugs, can we measure 7 Liters?

Or, having having 21 and 26 Liters water jugs, can we measure 3 Liters?

Water Jugs Problem

$$
\begin{aligned}
(0,0) &\to (a,0) && \text{fill 1st jug}\\
&\to (0,a) && \text{pour 1st jug into 2nd jug}\\
&\to (a,a) && \text{fill 1st jug}\\
&\to (2a-b,b) && \text{pour 1st jug into 2nd jug (assuming } 2a \geqslant b)\\
&\to (2a-b,0) && \text{empty 2nd jug}\\
&\to (0,2a-b) && \text{pour 1st jug into 2nd jug}\\
&\to (a,2a-b) && \text{fill 1st jug}\\
&\to (3a-2b,b) && \text{pour 1st jug into 2nd jug (assuming } 3a \geqslant 2b)
\end{aligned}
$$

At every step, there is an invariant! .
Amount of water in jugs at any step is a linear combination of a and b.

Lemma (Water Jugs)

Amount of water in each jug at any step is a linear combination of a and b.

Proof uses induction.

Greatest Common Divisor

Definition

A **common divisor** of a and b is a number that divides both a and b.

Example

6 is a common divisor of 60 and 72;
11 is a common divisor of 330 and 627;
19 is a common divisor 3070704 and 193325.

While 25 is not a common divisor of 425 and 120.

Definition

Greatest common divisor of two integers a and b is called the greatest common divisor of a and b, denoted by $\gcd(a, b)$.

Example

$\gcd(60, 72) = 12, \ \ \gcd(15, 32) = 1.$

Properties of GCD

1. $\gcd(n, n) = n, \quad gcd(n, 1) = 1, \quad \gcd(n, 0) = n.$
2. $\gcd(a, b) = \gcd(b, rem(a, b))$.

Proof of property 2.

Let c be an arbitrary common divisor of a and b.
Therefore, $a = ck_1$ and $b = ck_2$ for some k_1 and k_2. By Division Theorem,

$$a = bq + r, \text{ where } r = rem(a, b).$$
$$r = a - bq = ck_1 - ck_2q = c(k_1 - k_2q)$$

Thus, c is also a divisor of r, and consequently c is a common divisor of r and b. We have shown that any common divisor c of a and b, is also a common divisor of b and r.
Therefore, $\gcd(a, b) = \gcd(b, r)$. □

Euclid Algorithm

Greatest common divisor of two numbers can be computed using the **Euclid algorithm** which is based on relation:

$$\gcd(a, b) = \gcd(b, rem(a, b)).$$

Example

Find $gcd(1147, 899)$.

$$\begin{aligned}
\gcd(1147, 899) &= \gcd(899, 248) && \text{since} \quad rem(1147, 899) = 248 \\
&= \gcd(248, 155) && \text{since} \quad rem(899, 248) = 155 \\
&= \gcd(155, 93) && \text{since} \quad rem(248, 155) = 93 \\
&= \gcd(93, 62) && \text{since} \quad rem(155, 93) = 62 \\
&= \gcd(62, 31) && \text{since} \quad rem(93, 62) = 31 \\
&= \gcd(31, 0) && \text{since} \quad rem(62, 31) = 0 \\
&= 31
\end{aligned}$$

GCD as a Linear Combination

Theorem

The greatest common divisor of a and b is equal to the smallest positive linear combination of a and b.

Example

$\gcd(52, 44) = 4$ and therefore 4 can be written as a linear combination of 52 and 44:

$$\exists\, c, d \in \mathbb{Z} \quad \text{such that} \quad \gcd(52, 44) = 4 = c \cdot 52 + d \cdot 44$$

Indeed, $\gcd(52, 44) = 4 = 6 \cdot 52 + (-7) \cdot 44$

Corollary

An integer is linear combination of a and b iff it is a multiple of $\gcd(a, b)$.

GCD as a Linear Combination

Example

Consider $a = 259$ and $b = 70$.
Following the Euclid's Algorithm compute $\gcd(259, 70)$:

$$\begin{aligned}
\gcd(259, 70) &= \gcd(70, 49) & rem(259, 70) &= 49 & 259 &= 3 \cdot 70 + 49 \\
&= \gcd(49, 21) & rem(70, 49) &= 21 & 70 &= 1 \cdot 49 + 21 \\
&= \gcd(21, 7) & rem(49, 21) &= 7 & 49 &= 2 \cdot 21 + 7 \\
&= \gcd(7, 0) & rem(21, 7) &= 0 & 21 &= 3 \cdot 7 + 0 \\
&= 7
\end{aligned}$$

Thus, 7 can be written as a linear combination of 259 and 70.
In other words, there exist integers s and t such that

$$7 = s \cdot 259 + t \cdot 70.$$

How to find s and t?

Extended Euclid's Algorithm (Pulverizer)

Example

Consider $a = 259$ and $b = 70$. Use Extended Euclid's Algorithm, known as **Pulverizer**:

x	y	$rem(x, y) = x - q \cdot y$
259	70	$49 = a - 3 \cdot b$
70	49	$21 = b - 1 \cdot 49$
		$= b - 1 \cdot (a - 3 \cdot b)$
		$= -1 \cdot a + 4 \cdot b$
49	21	$7 = 49 - 2 \cdot 21$
		$= (a - 3 \cdot b) - 2 \cdot (-1 \cdot a + 4 \cdot b)$
		$= \boxed{3 \cdot a - 11 \cdot b}$
21	7	0

Water jugs puzzle

Problem

Is it possible to measure c liters using $a-$Liter and $b-$Liter jugs?

Solution

According to Water Jugs Lemma, amount of water in each jug at any step is a linear combination of a and b. But, any linear combination of a and b is a multiple of $\gcd(a, b)$. *Therefore, we can measure only an amount of water that is a multiple of* $\gcd(a, b)$.

Example

Can we measure 3 Liters using 21 and 26 Liter jugs?
We can measure any amount which is a multiple of $\gcd(21, 26) = 1$.
In other words, yes, we can measure 3 Liters.

Definition

Two integers a and b are called **relatively prime** if $\gcd(a, b) = 1$.

Water jugs puzzle

$$
\begin{array}{lllllll}
\xrightarrow{\text{fill 21}} & (21,0) & \xrightarrow{\text{pour 21 into 26}} & & & & (0,21) \\
\xrightarrow{\text{fill 21}} & (21,21) & \xrightarrow{\text{pour 21 to 26}} & (16,26) & \xrightarrow{\text{empty 26}} & (16,0) & \xrightarrow{\text{pour 21 to 26}} (0,16) \\
\xrightarrow{\text{fill 21}} & (21,16) & \xrightarrow{\text{pour 21 to 26}} & (11,26) & \xrightarrow{\text{empty 26}} & (11,0) & \xrightarrow{\text{pour 21 to 26}} (0,11) \\
\xrightarrow{\text{fill 21}} & (21,11) & \xrightarrow{\text{pour 21 to 26}} & (6,26) & \xrightarrow{\text{empty 26}} & (6,0) & \xrightarrow{\text{pour 21 to 26}} (0,6) \\
\xrightarrow{\text{fill 21}} & (21,6) & \xrightarrow{\text{pour 21 to 26}} & (1,26) & \xrightarrow{\text{empty 26}} & (1,0) & \xrightarrow{\text{pour 21 to 26}} (0,1) \\
\xrightarrow{\text{fill 21}} & (21,1) & \xrightarrow{\text{pour 21 to 26}} & & & & (0,22) \\
\xrightarrow{\text{fill 21}} & (21,22) & \xrightarrow{\text{pour 21 to 26}} & (17,26) & \xrightarrow{\text{empty 26}} & (17,0) & \xrightarrow{\text{pour 21 to 26}} (0,17) \\
\xrightarrow{\text{fill 21}} & (21,17) & \xrightarrow{\text{pour 21 to 26}} & (12,26) & \xrightarrow{\text{empty 26}} & (12,0) & \xrightarrow{\text{pour 21 to 26}} (0,12) \\
\xrightarrow{\text{fill 21}} & (21,12) & \xrightarrow{\text{pour 21 to 26}} & (7,26) & \xrightarrow{\text{empty 26}} & (7,0) & \xrightarrow{\text{pour 21 to 26}} (0,7) \\
\xrightarrow{\text{fill 21}} & (21,7) & \xrightarrow{\text{pour 21 to 26}} & (2,26) & \xrightarrow{\text{empty 26}} & (2,0) & \xrightarrow{\text{pour 21 to 26}} (0,2) \\
\xrightarrow{\text{fill 21}} & (21,2) & \xrightarrow{\text{pour 21 to 26}} & & & & (0,23) \\
\xrightarrow{\text{fill 21}} & (21,23) & \xrightarrow{\text{pour 21 to 26}} & (18,26) & \xrightarrow{\text{empty 26}} & (18,0) & \xrightarrow{\text{pour 21 to 26}} (0,18) \\
\xrightarrow{\text{fill 21}} & (21,18) & \xrightarrow{\text{pour 21 to 26}} & (13,26) & \xrightarrow{\text{empty 26}} & (13,0) & \xrightarrow{\text{pour 21 to 26}} (0,13) \\
\xrightarrow{\text{fill 21}} & (21,13) & \xrightarrow{\text{pour 21 to 26}} & (8,26) & \xrightarrow{\text{empty 26}} & (8,0) & \xrightarrow{\text{pour 21 to 26}} (0,8) \\
\xrightarrow{\text{fill 21}} & (21,8) & \xrightarrow{\text{pour 21 to 26}} & (3,26) & \xrightarrow{\text{empty 26}} & (3,0) & \xrightarrow{\text{pour 21 to 26}} (0,3)
\end{array}
$$

Prime numbers

Definition

An integer $p > 1$ is called **prime** if its only divisors are 1 and p.

Erastophenes Sieve can be used to find the first prime numbers

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Definition

An integer that is not prime (with exception of 0 and ± 1) is called **composite**.

Prime Numbers

	2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109
113	127	131	137	139	149	151	157	163	167
173	179	181	191	193	197	199	211	223	227
229	233	239	241	251	257	263	269	271	277
281	283	293	307	311	313	317	331	337	347
349	353	359	367	373	379	383	389	397	401
409	419	421	431	433	439	443	449	457	461
463	467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593	599
601	607	613	617	619	631	641	643	647	653
659	661	673	677	683	691	701	709	719	727
733	739	743	751	757	761	769	773	787	797
809	811	821	823	827	829	839	853	857	859
863	877	881	883	887	907	911	919	929	937
941	947	953	967	971	977	983	991	997	

Prime numbers distribution

Definition

$$\pi(n) = \big|\{p \in [2, \ldots, n] \mid p \text{ is prime } \}\big|.$$

In other words, $\pi(n)$ is the number of primes less or equal to n.
For example, $\pi(1) = 0, \quad \pi(2) = 1, \quad \pi(10) = 4$.

Prime Number Theorem

$$\lim_{n \to \infty} \frac{\pi(n)}{n/\ln n} = 1$$

Primes gradually become less dense. As a rule of thumb, about 1 integer out of every $\ln n$ in the neighbourhood of n is a prime number.
Prime Number Theorem was conjectured by Legendre in 1798 and proved a century later by Poussin and Hadamard in 1896.

Job Proposal

In late 2004 a billboard appeared in various locations around the country:

$$\boxed{\left\{\begin{array}{c}\text{first } 10-\text{digit prime found}\\ \text{in consecutive digits of } e\end{array}\right\}.\text{com}}$$

Substituting the correct number for the expression in curly-braces produced the URL for a Google employment page.
How hard is this problem?

$e = 2.7182818284590452353602874713526624977572470936$
$9995957496696762772407663035354759457138217852516642$
$7427466391932003059921817413596629043572900334295260595630738132328627943 4\ldots$

Would you have to look through thousands or millions or billions of digits of e to find a 10-digit prime?

The rule of thumb derived from the Prime Number Theorem says that among $10-$ digit numbers, about 1 in $\ln 10^{10} \approx 23$ is prime.

This suggests that the problem isn't really so hard!

$e = 2.71828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642742746639193200305992181741359662904357290033429526059563073813232862794 34\ldots$

Primality Testing

A simple search for factors of an integer n takes a number of steps proportional to $\sqrt{n}$, that is exponential in the size of n.

All known procedures for prime checking will grow fantastically fast as the size of input increases.

In 2002, an amazingly simple (description only 13 lines long), new method was discovered by *Agrawal*, *Kayal*, and *Saxena*, which showed that prime testing only required a polynomial number of steps $((\log n)^{12} steps)$.

Therefore, prime testing is definitely not in the category of infeasible problems requiring an exponentially growing number of steps in bad cases.

There are a lot of online implementations checking whether a number is prime or not.

Given the product of two large (really LARGE) primes $n = pq$, there is no efficient way to recover the primes p and q.

The best known algorithm is the **"number field sieve"**, which runs in time proportional to:

$$e^{1.9(\ln n)^{1/3}(\ln \ln n)^{2/3}}$$

This is infeasible when n has 300 digits or more.

Twin Prime Conjecture

There are infinitely many primes p such that $p + 2$ is also a prime.

Chen almost proved it in 1966 by showing that there are infinitely many primes p such that $p + 2$ is a product of at most 2 primes.

Goldbach Conjecture

Every even integer greater than 2 is equal to the sum of two primes.

Theorem (Fundamental Theorem of Arithmetic's)

Every positive integer n can be written in a unique way as a product of primes:

$$n = p_1 \cdot p_2 \cdot \ldots \cdot p_k, \quad p_1 \leqslant p_2 \leqslant \ldots \leqslant p_k.$$

In order to prove it we need some auxiliary results:

Lemma

If p is a prime and $p \mid a \cdot b$, then $p \mid a$ or $p \mid b$.

Proof.

The greatest common divisor of a and p must be either 1 or p, since these are the only positive divisors of p.
If $\gcd(a, p) = p$, then the claim holds, because a is a multiple of p.
Otherwise, $\gcd(a, p) = 1$ and so $p \mid b$ by a property of the GCD. □

Last Lemma by induction easily extends to the following result

Lemma

Let p be a prime. If $p \mid a_1 a_2 \dots a_k$, then p divides some a_i.

The proof of Fundamental Theorem of Arithmetic's follows from Well Ordering Principle and this lemma.

Beside solving "Entscheidungsproblem" and other important problems from the foundations of Computer Science, Alan Turing is supposedly the first who proposed to use number theory for encryption in 1937.

His work is still kept secret,
but let's guess what the Turing's cryptography approach might be.

One approach (that we will call Turing code 1.0) is first codify letters (words) in numbers. For example, replace each letter by a number: $A = 01, B = 02, C = 03, \ldots$

Thus, the word **"victory"** will become

v	i	c	t	o	r	y
22	09	03	20	15	18	25

Turing's code requires the message to be a prime number, so we may need to pad the result with a few more digits to make a prime.

v	i	c	t	o	r	y
22	09	03	20	15	18	25

In this case, appending the digits 13 gives the number

$$m = 2209032015182513,$$

which is prime. This is the message to be sent.

Beforehand The sender and receiver agree on a secret key, which is a large prime number k.

Encryption The sender encrypts the message m by computing:

$$m^* = m \cdot k$$

Decryption The receiver decrypts m by computing:

$$\frac{m^*}{k} = \frac{m \cdot k}{k} = m$$

For example, suppose that the secret key is the prime number

$$k = 22801763489$$

and the message m is "victory" written as

$$m = 2209032015182513$$

Then the encrypted message is:

$$\begin{aligned} m^* &= m \cdot k = 2209032015182513 \cdot 22801763489 \\ &= 50369825549820718594667857 \end{aligned}$$

Several questions will appear:

1. How can the sender and receiver ensure that m and k are prime numbers, as required?
2. Is Turing's code secure?

Questions about Turing's code 1.0

1. How can the sender and receiver ensure that m and k are prime numbers, as required?

Primality testing is a polynomial time problem. Still, a twelfth degree polynomial grows pretty fast, so the Agrawal, et al. procedure is of no practical use.

Still, good ideas have a way of breeding more good ideas, so there's certainly hope that further improvements will lead to a procedure that is useful in practice.

But the truth is, there's no practical need to improve it, since very efficient probabilistic procedures for prime-testing have been known since the early 1970's.

These procedures have some probability of giving a wrong answer, but their probability of being wrong is so tiny that relying on their answers is the best bet you'll ever make.

2. Is Turing's code secure?

The enemy sees only the encrypted message $m^* = m \cdot k$, so recovering the original message m requires factoring m^*.

Despite immense efforts, no really efficient factoring algorithm has ever been found.

In effect, Turing's code puts to practical use his discovery that there are limits to the power of computation.

Thus, provided m and k are sufficiently large, the enemy seem to be out of luck!

Breaking Turing's code 1.0

Let's consider what happens when the sender transmits a second message using Turing's code and the same key.

This gives the enemy two encrypted messages to look at:

$$m_1^* = m_1 \cdot k, \qquad m_2^* = m_2 \cdot k$$

The greatest common divisor of the two encrypted messages, m_1 and m_2, is the secret key k.

And the GCD of two numbers can be computed very efficiently.

So after the second message is sent, the enemy can recover the secret key and read every message!

Definition

Integer a is **congruent** to integer b modulo n iff $n \mid a - b$.
This is written $a \equiv b \pmod{n}$.

Example

$9 \equiv 6 \pmod{3}$ because $3 \mid 9 - 6$.

Example

$29 \equiv 15 \pmod{7}$ because $7 \mid 29 - 15$.

Example

$75 \equiv 21 \pmod{6}$ because $6 \mid 75 - 21$.

There is a close connection between congruence and remainders.

Lemma (Congruence and Remainders)

$a \equiv b \pmod{n}$ *iff* $rem(a, n) = rem(b, n)$.

Proof.

By the Division Theorem, there exist unique pairs of integers q_1, r_1 and q_2, r_2 such that:

$$a = q_1 n + r_1, \qquad 0 \leqslant r_1 < n$$
$$b = q_2 n + r_2, \qquad 0 \leqslant r_2 < n$$

Subtract equations:

$$b - a = (q_2 - q_1)n + (r_2 - r_1), \qquad -n < r_2 - r_1 < n$$

Now, $a \equiv b \pmod{n} \iff n \mid a - b \iff r_2 - r_1 = 0 \iff rem(a, n) = rem(b, n)$ □

Example

$9 \equiv 6 \pmod 3$ because $rem(9,3) = rem(6,3) = 0$.
Moreover, $9 \equiv 6 \equiv 3 \equiv 0 \pmod 3$.

Example

$29 \equiv 15 \pmod 7$ because $rem(29,7) = rem(15,7) = 1$.
Moreover, $29 \equiv 22 \equiv 15 \equiv 8 \equiv 1 \pmod 7$.

Example

$75 \equiv 21 \pmod 6$ because $rem(75,6) = rem(21,6) = 3$.
Moreover, $75 \equiv 69 \equiv 63 \equiv \cdots \equiv 27 \equiv 21 \equiv 15 \equiv 9 \equiv 3 \pmod 6$.

Corollary

$a \equiv rem(a, n) \pmod{n}$

Another way of looking at congruence is through relation "having the same remainder after division by n".

It is an equivalence relation (symmetric, reflexive and transitive).

Thus, all integers can be divided into partition classes. In one class we have all integers having the same remainder after division by n. For example, these are the partition classes if $n = 3$:

$$\{\ldots, -9, -6, -3, 0, 3, 6, 9, 12, \ldots\}$$
$$\{\ldots, -8, -5, -2, 1, 4, 7, 10, \ldots\}$$
$$\{\ldots, -10, -7, -4, -1, 2, 5, 8, \ldots\}$$

Lemma (Modular Arithmetic Lemma)

The following hold for $n \geq 1$*:*

1. $a \equiv a \pmod{n}$
2. $a \equiv b \pmod{n}$ *implies* $b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ *and* $b \equiv c \pmod{n}$ *implies* $a \equiv c \pmod{n}$
4. $a \equiv b \pmod{n}$ *implies* $a + c \equiv b + c \pmod{n}$
5. $a \equiv b \pmod{n}$ *implies* $ac \equiv bc \pmod{n}$
6. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *implies* $a + c \equiv b + d \pmod{n}$
7. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *implies* $ac \equiv bd \pmod{n}$

Let's fix number $n = 3$.

$$\{\ldots, -9, -6, -3, 0, 3, 6, 9, 12, \ldots\} = [0]_3$$
$$\{\ldots, -8, -5, -2, 1, 4, 7, 10, \ldots\} = [1]_3$$
$$\{\ldots, -10, -7, -4, -1, 2, 5, 8, \ldots\} = [2]_3$$

Note that $\forall n \in \mathbb{Z}$:

$$[0]_3 = [3]_3 = [6]_3 = [-3]_3 = [-6]_3 = \cdots = [3n]_3$$
$$[1]_3 = [4]_3 = [7]_3 = [-2]_3 = [-5]_3 = \cdots = [3n+1]_3$$
$$[2]_3 = [5]_3 = [8]_3 = [-1]_3 = [-4]_3 \cdots = [3n+2]_3$$

In other words, class can be represented by any of its elements.

Similarly, we can define classes modulo any integer n, denoted $[\cdot]_n$:

$$[0]_n,\ [1]_n,\ [2]_n,\ [3]_n,\ \ldots,\ [n-1]_n,$$

Operations in Modular Arithmetic

We can define operations of addition and multiplication in the set of classes modulo n.

Definition

Given $n \in \mathbb{Z}$ consider partition classes modulo n. Then,

$$[a]_n + [b]_n = [a+b]_n,$$
$$[a]_n \cdot [b]_n = [a \cdot b]_n$$

Example (Let $n = 3$)

$$[1]_3 + [1]_3 = [1+1]_3 = [2]_3,$$
$$[2]_3 + [1]_3 = [2+1]_3 = [0]_3,$$
$$[1]_3 \cdot [1]_3 = [1 \cdot 1]_3 = [1]_3,$$
$$[2]_3 \cdot [2]_3 = [2 \cdot 2]_3 = [1]_3,$$

$$[0]_3 + [1]_3 = [0+1]_3 = [1]_3,$$
$$[2]_3 + [2]_3 = [2+2]_3 = [1]_3,$$
$$[0]_3 \cdot [1]_3 = [0 \cdot 1]_3 = [0]_3,$$
$$[2]_3 \cdot [1]_3 = [2 \cdot 1]_3 = [2]_3.$$

Operations in Modular Arithmetic

Example (Let $n = 6$)

$$[3]_6 + [4]_6 = [1]_6, \quad [5]_6 + [5]_6 = [4]_6, \quad [2]_6 + [4]_6 = [0]_6,$$
$$[1]_6 \cdot [5]_6 = [5]_6, \quad [0]_6 \cdot [4]_6 = [0]_6, \quad [4]_6 \cdot [5]_6 = [2]_6.$$

Example (Let $n = 7$)

$$[2]_7 + [3]_7 = [5]_7, \quad [5]_7 + [5]_7 = [3]_7, \quad [6]_7 + [4]_7 = [3]_7,$$
$$[5]_7 \cdot [4]_7 = [6]_7, \quad [3]_7 \cdot [6]_7 = [4]_7, \quad [2]_7 \cdot [4]_7 = [1]_7,$$

Observe that we can perform the inverse operation of addition:

$$[4]_6 = [1]_6 - [3]_6, \quad [5]_6 = [4]_6 - [5]_6, \quad [2]_6 = [0]_6 - [4]_6,$$
$$[2]_7 = [5]_7 - [3]_7, \quad [5]_7 = [3]_7 - [5]_7, \quad [4]_7 = [3]_7 - [6]_7,$$

The set of classes modulo n together with addition and multiplication is called ring $\mathbb{Z}_n$.

Turing code version 2.0

Consider another (more secure?) version of Alan Turing's code.

The sender and receiver agree on a large prime p, which may be made public.
The secret key $k \in \{1, 2, \ldots, p-1\}$.

Encryption
The message m can be any integer $\in \{1, 2, \ldots, p-1\}$
(not necessary to be a prime).
The sender encrypts the message m to produce m^*:

$$m^* = rem(mk, p)$$

Decryption ???????????

Definition

The **multiplicative inverse** of a number x is number x^{-1} such that:

$$x \cdot x^{-1} = 1$$

Definition

The multiplicative inverse of an integer a modulo n is another integer, denoted a^{-1}, such that:

$$a \cdot a^{-1} \equiv 1 \pmod{n}$$

Example

$$2 \cdot 3 \equiv 1 \pmod{5} \quad \text{thus} \quad 2^{-1} \equiv 3 \pmod{5}$$
$$3 \cdot 3 \equiv 1 \pmod{4} \quad \text{thus} \quad 3^{-1} \equiv 3 \pmod{4}$$

Definition

The multiplicative inverse of an integer a modulo n is another integer a^{-1} such that:

$$a \cdot a^{-1} \equiv 1 \pmod{n}$$

$$[a]_n \cdot [a^{-1}]_n = [1]_n$$

Example

$$2 \cdot 3 \equiv 1 \pmod{5} \quad \text{thus,} \quad 2^{-1} \equiv 3 \pmod{5} \quad \text{or} \quad ([2]_5)^{-1} = [3]_5$$

$$3 \cdot 3 \equiv 1 \pmod{4} \quad \text{thus,} \quad 3^{-1} \equiv 3 \pmod{4} \quad \text{or} \quad ([3]_4)^{-1} = [3]_4$$

Does the multiplicative inverse always exist?
Try and find 2^{-1} modulo 4.

Lemma

If n is **prime**, *then any integer has multiplicative inverse modulo n.*

The sender and receiver agree on a large prime p, which may be made public. The secret key $k \in \{1, 2, \ldots, p-1\}$.

Encryption The message m can be any integer $\in \{1, 2, \ldots, p-1\}$ (not necessary to be a prime). The sender encrypts the message m to produce m^*:

$$m^* = rem(mk, p)$$

Decryption

$$m = rem(m^* k^{-1}, p)$$

where k^{-1} is the multiplicative inverse modulo p.

Decryption

$$m = rem(m^* k^{-1}, p)$$

where k^{-1} is the multiplicative inverse modulo p.
Indeed,

$$\begin{aligned} m^* \cdot k^{-1} &= rem(mk, p) \cdot k^{-1} \\ &\equiv (mk) \cdot k^{-1} \pmod{p} \\ &\equiv m(k \cdot k^{-1}) \pmod{p} \\ &\equiv m \cdot 1 \pmod{p} \\ &\equiv m \pmod{p}. \end{aligned}$$

In order to recover the message m, receiver computes the multiplicative inverse of k, that is k^{-1}, and finds the remainder of division of $m^* k^{-1}$ by p.

How to compute k^{-1}?

Turing code version 2.0

Multiplicative inverse of k modulo p can be computed using Extended Euclidean Algorithm (Pulverizer):

Since $s \cdot p + t \cdot k = 1$,

we have that $t \cdot k \equiv 1 \pmod{p}$

and therefore, $k^{-1} \equiv t \pmod{p}$

Thus, to decrypt a message encrypted with Turing 2.0 with key k,
we need to compute k^{-1} and

$$m = rem(m^* k^{-1}, p)$$

An alternative approach for computing multiplicative inverse is based on the following remarkable result:

Theorem (Fermat's Little Theorem)

Suppose p is a prime and k is not a multiple of p. Then:

$$k^{p-1} \equiv 1 \pmod{p}$$

Let p be a prime and k is not a multiple of p.
Then, by Fermat's Little Theorem we have

$$k^{p-2} \cdot k \equiv 1 \pmod{p}.$$

Thus k^{p-2} is a multiplicative inverse of k.

Example (6^{-1} modulo 17)

Using Fermat's Little Theorem, we have

$$6^{15} \cdot 6 \equiv 1 \pmod{17},$$

in other words, we need to compute $rem(6^{15}, 17)$.

$$\begin{aligned}
6^2 &= 36 \equiv 2 \pmod{17} \\
6^4 &= (6^2)^2 \equiv 2^2 \pmod{17} \equiv 4 \pmod{17} \\
6^8 &= (6^4)^2 \equiv 4^2 \pmod{17} \equiv 16 \pmod{17} \\
6^{15} &= 6^8 \cdot 6^4 \cdot 6^2 \cdot 6 \equiv 16 \cdot 4 \cdot 2 \cdot 6 \pmod{17} \equiv 3 \pmod{17}.
\end{aligned}$$

$$\begin{aligned}
3 \cdot 6 &\equiv 1 \pmod{17} \\
6^{-1} &= 3 \pmod{17}.
\end{aligned}$$

Turing code version 2.0

Unfortunately, Turing 2.0 can be broken by so-called **plain-text attack**.

Suppose that both clear-text m and cipher-text m^* are known, where

$$m^* = rem(mk, p).$$

Then,

$$\begin{aligned} m^{p-2} \cdot m^* &= m^{p-2} \cdot rem(mk, p) \\ &\equiv m^{p-2} \cdot mk \pmod{p} \\ &\equiv m^{p-1} \cdot k \pmod{p} \\ &\equiv 1 \cdot k \pmod{p} \\ &\equiv k \pmod{p}. \end{aligned}$$

Therefore, if adversary intercepts both m and m^*, then, by computing $m^{p-2} \cdot m^*$ and dividing the result by p, the remainder will provide the secret key k.

Definition

$a \equiv b \pmod{n}$) if and only if $n \mid a - b$.
$a \equiv b \pmod{n}$ if and only if $rem(a, n) = rem(b, n)$.

The set of integers $\mathbb{Z}$ can be partitioned into n equivalence classes:

$$[0]_n,\ [1]_n,\ [2]_n, \ldots,\ [n-1]_n$$

and we can define arithmetic operations (addition and multiplication) on these classes.

Definition

The multiplicative inverse of an integer a modulo n is integer a^{-1},

$$a \cdot a^{-1} \equiv 1 \pmod{n},$$
$$[a]_n \cdot [a^{-1}]_n = [1]_n.$$

Lemma

If n is **prime**, *then any integer has multiplicative inverse modulo n.*

Multiplicative inverse of k modulo p can be computed using one of these methods:

1. Extended Euclidean algorithm:

$$k^{-1} \equiv t, \text{ where } s \cdot p + t \cdot k = 1 = gcd(p, k).$$

2. Little Fermat's Theorem: Let p be a prime and k be not a multiple of p. Then:

$$k^{p-1} \equiv 1 \pmod{p} \quad \Rightarrow \quad k^{-1} = k^{p-2}.$$

RSA cryptosystem: historical glance

In 1977 the internet, electronic mail, electronic banking were in their infancy. Major obstacle: security and privacy.

There had to be some way to send messages across phone lines without unwanted eavesdroppers being able to intercept and understand them.

Public Key Cryptography — concept proposed by Diffie and Hellman, 1976.

Its novelty consisted in:

Making the encryption key public did not reveal the decryption key!

RSA algorithm was proposed in 1977 by Ronald **R**ivest, Adi **S**hamir, Len **A**dleman.

Prior to RSA people would encipher their messages and send them with a courier to the recipient of the message together with the key to decipher it.

Public Key Cryptosystems

Let M be the set of all possible messages and K be the set of all "keys".

For each key $k \in K$ there exists both a decryption function

$$D_k : M \to M$$

and an encryption function

$$E_k : M \to M.$$

In order to be considered a public key cryptosystem these functions must satisfy the following conditions:

1. $\forall m \in M$ and $\forall k \in K$, $E_k(D_k(m)) = m$ and $D_k(E_k(m)) = m$.
2. $\forall m \in M$ and $\forall k \in K$, the values of $E_k(m)$ and $D_k(m)$ are not difficult to compute.
3. For almost every $k \in K$ if somebody knows only the function E_k, it is computationally infeasible to compute D_k.
4. Given $k \in K$, it is easy to find the functions E_k and D_k.

A function E_k that satisfies conditions $(1-4)$ is called a *trap-door one-way permutation.*

one-way — easy to compute in one direction but not in the other

trap-door — inverse function become easy to compute once certain information is revealed

permutation — every message is an encryption of another message, and every encrypted message is also a permissible message (useful in "signing" electronic documents)

Example

Consider two people: John and Maria.

Their encryption and decryption keys are $E_J, D_J,$ and E_M, D_M.

Their encryption keys E_J and E_M are public.

Their decryption keys D_J and D_M are secret.

John sends a message m to Maria by recovering her encryption key from her public file and encrypting his message with it.

He sends Maria $E_M(m)$ and only she can decipher it since only she knows D_M.

Thus she reads message $D_M(E_M(m)) = m$.

Electronic Signature

RSA can be used to send electronic signatures.

Example

If John is sending the message to Maria, he signs it by computing $S = D_J(m)$.

He then encrypts it in a usual way using E_M from Maria's public file.

The message received by Maria is $E_M(S)$, which Maria can dercypt with her own private decryption to get S.

Since Maria expects a message from John she knows to extract the original message using the John's public encryption key E_J : and she gets $E_J(S) = m$. The result is the message and signature pair: (m, S).

There are several candidates for a one way function. One of them is factoring.

Factoring. The function $f : (x, y) \to xy$

It is conjectured to be an one way function. The asymptotically proven fastest factoring algorithms to date are variations on Dixon's random squares algorithm.

It is a randomized algorithm with running time $L(n)^{\sqrt{2}}$ where $L(n) = e^{\sqrt{\log n \log\log n}}$.

The number field sieve by Lenstra, Manasee, and Pollard with modifications by Adlemann and Pomerance is a factoring algorithm proved under a certain set of assumptions to factor integers in expected time

$$e^{((c+o(1))(\log n)^{1/3}(\log\log n)^{2/3})}.$$

Euler function

Definition

Let $\varphi(n)$ denotes the number of integers smaller than n that are relatively prime with n.
Function $\varphi(n)$ is called the **totient function** or **Euler function**.

Example

$\varphi(6) = 2$, since 1 and 5 are relatively prime with 6.

$\varphi(12) = 4$, since $1, 5, 7$ and 11 are relatively prime with 12.

Example

If p is a prime number, then $\varphi(p) = p - 1$.
(since all numbers $\leqslant p$ are relatively prime to p.

Theorem

For any number n, if $p_1, p_2, \ldots, p_j$ are distinct prime factors of n, then

$$\varphi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\left(1-\frac{1}{p_3}\right)\ldots\left(1-\frac{1}{p_j}\right).$$

Example

$$\begin{aligned}\varphi(300) &= \varphi(2^2\cdot 3\cdot 5^2)\\ &= 300\left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)\left(1-\frac{1}{5}\right)\\ &= 300\cdot\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{4}{5}\\ &= 80.\end{aligned}$$

Corollary

If $n = pq$ with p and q prime numbers, then

$$\varphi(n) = (p-1)(q-1).$$

Proof.

Let $n = pq$. Then according to the last theorem we have

$$\phi(n) = pq\left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = pq\frac{p-1}{p}\frac{q-1}{q} = (p-1)(q-1).$$

□

Theorem (Euler's Theorem)

Suppose that n is a positive integer and k is relatively prime to n. Then

$$k^{\varphi(n)} \equiv 1 \pmod{n}.$$

RSA Algorithm

In order to encrypt a message with RSA, one must choose two large prime numbers p and q, usually of about 50 digits each.

Let $n = p \cdot q$.

The encryption key is a pair of integers (e, n) and the decryption key is a pair (d, n).

Given a message m, in order to encrypt it, we would represent it as a number between 0 and $n - 1$.

If the message is too large, break it in blocks, as long as every block is between 0 and $n - 1$.

Then

$$E(m) = m^e \equiv m^* \pmod{n} \quad \Leftrightarrow \quad m^* = rem(m^e, n)$$

and

$$D(m^*) = (m^*)^d \equiv m \pmod{n} \quad \Leftrightarrow \quad m = rem((m^*)^d, n)$$

How to choose e and d, such that the above formulas will work?

Integers e and d are closely related to numbers p and q.

Choose e to be any large random integer that is relatively prime to $(p-1)(q-1)$.

Then d will be the multiplicative inverse of e modulo $(p-1)(q-1)$:

$$e \cdot d \equiv 1 \pmod{(p-1)(q-1)}.$$

Theorem

The RSA cryptosystem satisfies : $E(D(m)) = m$, $D(E(m)) = m$.

Proof.

Numbers e and d are chosen such that

$$e \cdot d \equiv 1 \pmod{(p-1)(q-1)}.$$

So, there exists an integer r such that

$$e \cdot d = 1 + r(p-1)(q-1).$$

Need to show that $\boxed{D(E(m)) = m}$, i.e. $(m^*)^d \equiv m \pmod{n}$.

On the other hand,

$$\begin{aligned}(m^*)^d &\equiv m^{ed} \pmod{n} \\ &\equiv m^{1+r(p-1)(q-1)} \pmod{n} \\ &\equiv m \cdot m^{r(p-1)(q-1)} \pmod{n}.\end{aligned}$$

Proof.

By Euler's Theorem and the assumption that $\gcd(m, n) = 1$, we know that

$$m^{\varphi(n)} \equiv 1 \pmod{n}.$$

Since

$$\varphi(n) = \varphi(pq) = (p-1)(q-1),$$

we have

$$m^{\varphi(n)} = m^{(p-1)(q-1)} \equiv 1 \pmod{n}.$$

Hence,

$$\begin{aligned}(m^*)^d &\equiv m \cdot m^{r(p-1)(q-1)} \pmod{n} \\ &\equiv m \cdot \left(m^{(p-1)(q-1)}\right)^r \pmod{n} \\ &\equiv m \cdot 1^r \pmod{n} \\ &\equiv m \pmod{n}.\end{aligned}$$

RSA Example

Example

Let $p = 47$ and $q = 59$.

Then $n = 47 \cdot 59 = 2773$.

Choose $d = 157$.

Then $\varphi(2773) = 46 \cdot 58 = 2668$.

By Extended Euclidean algorithm we can find multiplicative inverse of d modulo 2668: $e = 17$. So, $157 \cdot 17 \equiv 1 \pmod{2668}$.

Suppose we have a message

$$M = \text{IT'S ALL GREEK TO ME}$$

Set: blank=00, $A = 01$, $B = 02$, $C = 03$, $D = 04$, $E = 05$, $F = 06$, ...

The encoded message will be

$$m = 0920\ 1900\ 0112\ 1200\ 0718\ 0505\ 1100\ 2015\ 0013\ 0500.$$

Example

Let m_1 be the first block. To encrypt it we compute

$$E(m_1) = (m_1)^{17} \equiv (920)^{17} \equiv 948 \pmod{2773}.$$

The final encrypted message is

$$E(m) = c = 0948\ 2342\ 1084\ 1444\ 2663\ 2390\ 0778\ 0774\ 0219\ 1655.$$

It is easy to check that decryption works. For m_1 :

$$(948)^{157} \equiv 920 \pmod{2773}.$$

Security of RSA

Security of RSA system is not perfect!!!

It rests on the difficulty of factoring n.

Recall that encryption key is public. Therefore, integers e and n are known. But in order to find d, someone needs to factor n to get p and q.

There are different factoring algorithms. Based on them here are a few recommendations for increased security:

1. p and q should be of slightly different sizes and d should be large in order to guard against any particular attack.
2. Choose p such that most of the digits are not predictable. If we choose our p by testing numbers for primality of the form $N \cdot 10^{50} + k$ for a random 50-digit number N and $k = 1, 3, 5, \ldots$ then the attacker can easily compute 47 of the last 50 digits.
3. If e is large enough, then d is computed with greater difficulty.

How hard is it to invert RSA?

We know that if we can factor n, then we can invert RSA via the Chinese Remainder Theorem, however we don't know if the converse is true.

Thus far, the best way known to invert RSA is to first factor n.

There are a variety of algorithms for this task. The best running time for a fully proved algorithm is Dixon's random squares algorithms which runs in time

$$O\left(e^{\sqrt{\ln n \ln \ln n}}\right).$$

In practice we may consider others.

Let $L = |p|$ where p is the smallest prime divisor of n. The Elliptic Curve algorithm takes expected time

$$O\left(e^{\sqrt{2L \ln L}}\right).$$

The Quadratic Sieve algorithm runs in expected

$$O\left(e^{\sqrt{\ln n \ln \ln n}}\right).$$

Notice the difference in the argument of the super-polynomial component of the running time. This means that when we suspect that one prime factor is substantially smaller than the other, we should use the Elliptic Curve method, otherwise one should use the Quadratic sieve.

The new number field sieve algorithm seems to achieve a

$$O\left(e^{1.9(\ln n)^{1/3}(\ln n \ln)^{2/3})}\right).$$

running time which is a substantial improvement asymptotically although in practice it still does not seem to run faster than the Quadratic Sieve algorithm for the size of integers which people currently attempt to factor.

The recommended size for n these days is 1024 bits.

Summary

- Private key encryption
  - Substitution cipher, Caesar cipher, One time pad
- Security requirements and properties
- Number Theory
  - Divisibility properties
  - Greatest Common Divisor, Euclid algorithm
  - GCD as a linear combination, Extended Euclid algorithm
  - Primality, Factoring, Fundamental Theorem of Arithmetics
  - Turing code 1.0
  - Modular arithmetic, Operations in modular arithmetic
  - Turing code 2.0
- Public key encryption
- RSA algorithm