Cryptography

(Originally for ACCESS)

E. Chamberlain, F. Guevara Vasquez, C. Hohenegger, and N. Korevaar

June 30, 2014

1 Introduction

Cryptography or the art of transmitted hidden messages has been long associated for armed conflicts between countries. For example, the Spartans were warned of an incoming invasion by the Persian fleet via a message written on the wood beneath a wax tablet. There are other means some more complex than others to hide a message. However, if the message is discovered, then the secrecy is lost. One way to prevent the message from being discovered is to transpose the letters like an anagram. However, this is not very secure and since the ancient times, people have been trying to come up with clever way to scramble (encrypt) a message, so that only the intended receiver can decipher (decrypt) it.

Definition 1.1. The **plain alphabet** is the alphabet (English, French, Spanish, ...) used to write the original message. The **cipher alphabet** is the alphabet of the letters that are substituted in place of plain letters.

One of the earliest example of mono-alphabetic substitutions ciphers are Caesar shifts.

Example 1.1. In a Caesar shift, the alphabet is simply shifted by an agreed upon amount.

Plain	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
Cipher	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Plain	t	u	v	w	x	y	z												
Cipher	A	B	C	D	E	F	G												

If you want to learn more about the history of cryptography, the *Code Book* by Simon Singh is a great reference.

In 1918 Scherbius invented his Enigma machine (a complicated electrical set of scramblers), and the German military quickly realized the importance of such a formidable machine. At that time, communication was via landlines and airwaves. As a first act of war, the British destroyed the German cables running in the Atlantic, forcing Germany to use airwaves to communicate. Furthermore, tremendous resource (money and men) were invested to break the Enigma. Finally, Alan Turing's machine helped in figuring out that supposedly unbreakable code, and hastened the end of the war. At the same time, the American Navy decided to use Navajo code talkers to send coded messages, since the Navajo language was like no other (especially impossible for both Japan and Germany to break). Unfortunately

	0	1	2	3	4	5	6	7	8	9
0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1	SP	A	B	C	D	E	F	G	H	I
2	J	K	L	M	N	O	P	Q	R	S
3	T	U	V	W	X	Y	Z	a	b	c
4	d	e	f	g	h	i	j	k	l	m
5	n	o	p	q	r	s	t	u	v	w
6	x	y	z	.	,	:	;	’	”	‘
7	!	@	#	$	%	ˆ	&	*	-	+
8	(	)	[	]	{	}	?	\	<	>
9	0	1	2	3	4	5	6	7	8	9

Table 1: Davis conversion table

many code talkers got killed by friendly fire in the Pacific.

As computers became more commonplace, code creators adapted. Since computers only deal with strings of 0's and 1's (binary), each letter in a message is replaced by its ASCII (American Standard Code for Information Interchange) binary number, creating long strings of numbers which can be scrambled and otherwise manipulated, sent, and then descrambled and read. Instead of using the ASCII conversion (base 2), we use the Davis conversion table in base 10 for our purposes.

The entries with XX are not used and "SP" is the space character. For example, the letter "K" is 21, the number "4" is 94.

Example 1.2. Use the Davis table to convert the two word sentence *Hello there!* into two number packets, each of which is less than 10^{12} (since each character is represented by two digits).

By the mid 1970's there were amazingly complicated encryption algorithms which could be made essentially unbreakable. For example, Horst Feistel's Lucifer cipher encrypts messages according to a scrambling operation and it can be set up with large enough keys so to be secure. A version which is small enough for the U.S. National Security Agency (NSA) to crack became the widely-used DES (Data Encryption Standard).

However, up until this point, no matter how convoluted the the encryption methods were, and how frequently the keys were changed for security reasons, all methods required that both parties to the message possessed *the* key for encryption and decryption. It was just assumed, because this had always been the case, that if you possessed the method to encrypt a message, then this was equivalent after some work, to also knowing how to decrypt it.

As the precursor to the internet, namely the ARPAnet, was beginning to grow, Whitfield Diffie and Martin Hellman, as well as others, realized the huge potential for electronic transactions, together with the need for assured security. Diffie-Hellman were perhaps the first to realize that there was an entirely new way to think of cryptography; that perhaps there were encryption keys which you could let everyone in the world know, but for which you could never the less keep secret the decryption key. This would solve the problem of key distribution, since if you wanted to receive secure messages you could tell the world how to encrypt anything they wanted to send you, but only you would know the decryption key which could stay safely at home. Diffie-Hellman called such encryption keys, "trapdoor", or "one way" functions, because you could make your personal encryption function public, and be essentially certain that no one would figure out how to find the (inverse) decryption function. This method of passing secret messages is called **public key cryptography** and is why it is possible to complete secure transactions on the internet, among other uses. In 1977, Ronald Rivest, Adi Shamir and Leonard Adleman described one of the easiest one-way functions, and the resulting method of public key cryptography is called RSA, in their honor. As we shall see, this method relies on number theory and modular (aka clock, remainder, or residue) arithmetic.

2 Modular artithmetic

2.1 Definitions

Definition 2.1. Let n be a positive integer divisor, and let a be any integer. Let a divided by n have quotient integer q, with a remainder r, $0 \leq r < n$. In other words, let

$$a = qn + r.$$

Then the notation for the **remainder** r is

$$a \mod n$$

and this remainder is called **the residue of** a **mod** n. The divisor n is called the **modulus** or the **clock number**.

Example 2.1. People in the military use a 24-hour clock for telling time. So 3:00 in the morning is 3:00, but 3:00 in the afternoon is 15:00. If you go to a military base and are told to be on the practice field at 18:00, what time (on a 12-hour clock) should you be there?

Example 2.2. If it is 10 am and Josh is picking you up in 7 hours, assuming he is on time, what time will he be there (In military time and standard time) ?

These are simple problems. But what is really going on here? We add the numbers together, but we don't care about how many revolutions are made, just about what is left

over. For small numbers like these, we see that it is easy enough to add and figure out the correct number, but for large numbers can you figure out a fast way to do this?

Example 2.3. If it is 5:00 and you have to leave for the airport in 39 hours, what time do you need to leave (military time)?

Example 2.4. If it is 8:00, and you have an appointment in 1984609 hours, what time is your appointment (military time)? You can use your calculator.

Example 2.5. What is 1984609 mod 24? Hint: use your work from the previous example.

Example 2.6. The goal is to give a formula for a 4-letter Caesar shift.

- Assign a number value to each letter in the alphabet according to the table below.
- For each letter in the message, replace it with its number value. Put each of those values in our encrypting function $f(x) = x + 4$ but cycle around the alphabet when you need to. For example, $f(23) = 23 + 4 = 27$ which is the same as $1 = B$. Find the corresponding letter for the new numbers. Write the encrypting function using mod .
- Send the message "REPLY" to someone using this Caesar shift. What is your encrypted message?
- What is a formula for the decryption function?

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

Remark 2.1. Caesar shifts are definitely not suitable for public key cryptography, because if someone tells you the encryption function you can easily deduce the decryption function; Caesar shifts are "clock" addition on the residue numbers **mod** 26, and the decryption function is the corresponding clock subtraction. The other drawback is that Caesar shifts are just a special case of mono-alphabetic substitution ciphers, which frequency analysis solves. If we grouped letters (and their corresponding numbers) together into packets, then we could use clock numbers (moduli) hugely bigger than 26, but it would be just as easy as before to figure out the decryption function from the encryption function.

We'll see that besides addition, you can also multiply and take powers in clock (modular) arithmetic. It will turn out that certain power functions are effective one-way functions for moduli which are products of two huge primes, and this is the basis for RSA cryptography. To to understand RSA cryptography we're first going to have to get really good at modular addition and multiplication.

2.2 Addition and Multiplication

We need a little more notation.

Definition 2.2. Let n be a positive integer, and let a and b be integers. We write

$$a \equiv b \mod n$$

when a and b differ by a multiple of n, i.e. when $a - b$ is a multiple of n, or equivalently a equals b plus a multiple of n. We say a **is congruent to** b **mod** n.

Notice that $a \equiv b \mod n$ means that a and b have the same residue $\mod n$. In particular a is congruent to its residue $a \mod n$.

Modular addition and multiplication are just regular integer addition and multiplication, except that we only care about clock location, so we only need to keep track of whether sums or products are congruent $\mod n$. Here are some important properties related to modular arithmetic.

Proposition 2.1. *If a, b, and n are integers, and if $a \equiv b \mod n$, then $b \equiv a \mod n$.*

Pf. If $a - b$ is a multiple of n, then $b - a$ is the opposite multiple of n. □

Proposition 2.2. *If a, b, and n are integers, and if $a \equiv b \mod n$ and $c \equiv d \mod n$, then $a + c \equiv b + d \mod n$.*

Pf. If $b = a + kn$ and $d = c + ln$, then $b + d = a + c + (k + l)n$. □

Proposition 2.3. *If a, b, and n are integers, and if $a \equiv b \mod n$ and $c \equiv d \mod n$, then the products $ac \equiv bd \mod n$.*

Proof. If $b = a + kn$ and $d = c + ln$, then $bd = (a + kn)(c + ln) = ac + (kc + al + kln)n$. □

Example 2.7. We know that $17 \equiv 2 \mod 5$ and $14 \equiv 4 \mod 5$. Find

$$17 + 14 \mod 5, \quad (17)(14) \mod 5, \quad 17^3 \mod 5,$$

using the properties above to save work.

Complete the following exercises in groups.

Exercise 2.1. Find the residue x which solves the equation $x - 8 \equiv 9 \mod 13$.

Exercise 2.2. List all of the integers x between -50 and 50 which satisfy $x \equiv 7 \mod 17$.

Exercise 2.3. Fill in the missing residue value: $-3 \equiv$ ____ $\mod 11$.

Exercise 2.4. Find residue values for $21 + 83 \mod 5$, and for $(21)(83) \mod 5$ efficiently.

Exercise 2.5. Find all of the integers y between 1 and 100 which satisfy $y \equiv 13 \mod 20$.

Exercise 2.6. Fill in the missing residue numbers:

1. $19 \equiv$ ____ $\mod 6$
2. $20568 \equiv$ ____ $\mod 19$
3. $-39 \equiv$ ____ $\mod 16$

Exercise 2.7. Solve for x in the equation $3 - x \equiv 7 \mod 8$.

Exercise 2.8. Find the residue numbers for $7^5 \mod 9$, and for $2^{10} \mod 7$, efficiently. Hint: Group the terms, think of 3^4 as $(3)(3)(3)(3)$.

Exercise 2.9. Find the missing residue number x, if it exists:

1. $3x \equiv 5 \mod 8$
2. $2x \equiv 5 \mod 8$

2.3 Functions

Remark 2.2. In Example 2.6 we used

$$f(x) = x + 4 \mod 26$$

to encrypt a message. The domain and range for this function is the collection or residue numbers $\{0, 1, 2, \ldots 25\} \mod 25$, which we have identified with the 26 letter alphabet. Our function described a Caesar shift by 4 letters. The inverse (decryption) function is

$$g(x) = x - 4 \mod 26.$$

We will eventually be encrypting long packets of numbers corresponding to long strings of letters and punctuation, and we will be using power functions in modular arithmetic with very large moduli. In this section, we're focusing on addition an multiplication, and encryption functions, like the example above, which use these two operations. Our example moduli will be small numbers we can work with by hand.

Example 2.8. For small moduli it's sometimes helpful to use addition or multiplication tables. Here is the addition table for modulus 5:

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Note that if n were large it would not be profitable to make a huge addition table.

Exercise 2.10. Suppose we had a function $f(x) = x + 2 \mod 5$. Compute the following:

1. $f(3)$
2. $f(1)$

3. $f(2)$

What row(s) of the addition table could you look at to show how $f(x)$ permutes the residue values?

Exercise 2.11. Now suppose we are given that $g(x) = x - 2 \mod 5$. (The inverse or "undo" function of $f(x)$.) Using the table, compute the following:

1. $g(0)$

2. $g(3)$

3. $g(4)$

Exercise 2.12. Can you think of another formula, one which uses addition rather than subtraction, that yields the same inverse function $g(x)$? Can you illustrate how $f(x)$ and $g(x)$ are inverse functions, using the addition table?

Remark 2.3. There are some subtleties happening with $g(x)$. How did we find $g(x)$? Simple, we just needed to find out how to undo whatever happened in $f(x)$. Since we added 2 to our value in $f(x)$, then we would just need to subtract 2 (or add -2) to get $g(x)$. What we are really doing is finding the additive inverse for 2.

Definition 2.3. The **additive inverse** of an integer a is a number b such that $a + b \equiv 0$.

Exercise 2.13. Find the residue numbers which are additive inverses of the following:

1. $3 \mod 39$

2. $18 \mod 56$

3. $-4 \mod 20$

Example 2.9. Find the residue number solution x to the equation $3 - x \equiv 7 \mod 8$.

Solution. We solve this equation the same way we would solve $3 - x = 7$. If we subtract 3 from both sides of this equation we get $-x \equiv 4 \mod 8$. Multiply by -1 to get $x \equiv -4 \mod 8$. Thus $x \equiv -4 \mod 8$, so $x \equiv 4 \mod 8$. □

Exercise 2.14. Find a residue number solution x for $7 - x \equiv 21 \mod 24$.

Now lets consider multiplication, and multiplication functions. Here is the multiplication table for modulo 5. Make sure to check some of the entries to see that you agree:

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Exercise 2.15. Let $f(x) = 2x \mod 5$. Compute the following:

1. $f(3)$

2. $f(1)$

3. $f(2)$

What is the inverse of this function? Is it $g(x) = \frac{x}{2}$? If it were then $g(1) = \frac{1}{2}$ which is not possible. So how do we find $g(x)$? To answer this question we need to find the multiplicative inverse of 2.

Definition 2.4. The **multiplicative inverse** of an integer a is a number c such that $ac \equiv 1$.

Now we can look at our multiplication table to find the multiplicative inverse of 2, which we see is 3.

Exercise 2.16. Compute the following with $g(x) = 3x \mod 5$ using the table.

1. $g(1)$

2. $g(2)$

3. $g(4)$

Exercise 2.17. Using the same table as in Example 2.6 (repeated below), encrypt the message "ATTACK AT DAWN" using the function $f(x) = 5x \mod 26$.

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

Exercise 2.18. Can you find the inverse function needed to decrypt your message from the previous exercise?

Example 2.10. Fill in the mod 15 multiplication table, and find the multiplicative inverses.

×	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	0	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	0	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	0	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7															
8															
9															
10	0	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	0	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	0	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	0	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	0	14	13	12	11	10	9	8	7	6	5	4	3	2	1

a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
b															

What do you notice ?

Exercise 2.19. List the numbers which have inverses. How do these numbers relate to 15?

Exercise 2.20. List the numbers which do not have inverses. How do these numbers relate to 15?

Exercise 2.21. What do you notice about row a when a has a multiplicative inverse, as compared to when it doesn't? In rows where the pattern of products repeats, how many times does it repeat, and when does the first repetition occur?

Definition 2.5. A **one-way function** is a function that is easy to compute on every input, but hard to invert given the image of a randome input.

Remark 2.4. One-way functions are the mathematical constructs representing the encoding of the message. Encoding should be easy, deciphering the message for someone other than the intended receiver should be hard.

Example 2.11. If we use the encryption function $f(x) = x + a \mod N$ to permute the residue numbers, corresponding to Caesar shifts for number of packets, anyone who understands clock arithmetic can deduce the decryption function $g(x) = x - a \mod N$ immediately. Modular addition is NOT a one-way function.

Example 2.12. We can try using modular multiplication as a one-way function, $f(x) = ax \mod N$. For f to be a good encryption function, we need its inverse function g to exist and to be hard to find for an attacker. From the previous examples, for the multiplicative inverse to exist, we need to have $\gcd(a, N) = 1$. We will prove this statement in Section 3. We might think that if we take N large enough, finding the multiplicative inverse of a is a hard problem because we have to list many numbers. This turns out to be false and in Section 4, we will use Euclid's algorithm to quickly find a multiplicative inverse b of $a \mod N$. Then $g(x) = bx \mod N$ is the decryption function. Again modular multiplication is NOT a one-way function.

Example 2.13. A power function of the form $f(x) = x^e \mod N$ for "good choices" of e, N is a one-way function. Such functions form the basis of RSA cryptography.

2.4 Powers

Example 2.14. Let $N = 11$ and let our candidate encryption function be $f(x) = x^2 \mod 11$. Complete the table below. Does a decryption function g exist ?

x	0	1	2	3	4	5	6	7	8	9	10
x^2	0	1	4	9	5						

Example 2.15. Keeping $N = 11$, show that the function $f(x) = x^3 \mod 11$ does encrypt (permute) the residue numbers:

x	0	1	2	3	4	5	6	7	8	9	10
x^3	0	1	8								

Remark 2.5. We might hope, based on our experiences with addition and multiplication encryption functions, that if our encryption function is a power function $f(x) = x^e \mod N$, then our decryption function is $g(x) = x^d \mod N$, for some power d.

Example 2.16. For the function $f(x) = x^e \mod N$ with $e = 3$ and $N = 11$, find the decryption function $g(x) = x^d \mod N$, in other words find the power d. Since $f(2) = 8 \mod 11$, we want $g(8) = 2$, i.e. $8^d \equiv 2 \mod 11$. Compute successive powers of 8 until you are able to solve this equation for d.

But we need to check that the decryption power $d = 7$ works for every x in our residue range! Let's divide the work among gourps.

Exercise 2.22. Let group number x check that this is true for the residue number x, with the following exceptions: Since $x = 1$ is immediate, and since we just checked $x = 2$, Group 1 should check $x = 8$ and Group 2 should check $x = 9$. ($x = 10 \equiv -1 \mod 11$ is also easy.) Each groups wants to check that

$$g(f(x)) \equiv x \mod N \Leftrightarrow f(x)^d \equiv x \mod N.$$

Be clever to minimize your computing! The easy way to fill in a row is to multiply previous entries by the residue number for that particular row, $\mod 11$.

Exercise 2.23. Since RSA cryptography uses moduli $N = pq$, where p and q are prime numbers, we'll experiment with small prime numbers $p = 3$, $q = 5$, $N = 15$, and use the mod 15 table of powers below to figure out good and bad encryption powers e. (A good encryption function permutes the residue numbers, so that it has an inverse decryption function.) First, fill in rows 6 and 7 of the table!

power →	1	2	3	4	5	6	7	8	9	10
residue										
0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1
2	2	4	8	1	2	4	8	1	2	4
3	3	9	12	6	3	9	12	6	3	9
4	4	1	4	1	4	1	4	1	4	1
5	5	10	5	10	5	10	5	10	5	10
6										
7										
8	8	4	2	1	8	4	2	1	8	4
9	9	6	9	6	9	6	9	6	9	6
10	10	10	10	10	10	10	10	10	10	10
11	11	1	11	1	11	1	11	1	11	1
12	12	9	3	6	12	9	3	6	12	9
13	13	4	7	1	13	4	7	1	13	4
14	14	1	14	1	14	1	14	1	14	1

Exercise 2.24. $f(x) = x^3 \mod 15$ is a good encryption function. What part of the power table confirms this fact? Find a power d so that $g(x) = x^d \mod 15$ is the decryption function for $f(x)$. Use the power table to check your work.

3 Number theory

We've been doing a lot of experimentation with modular arithmetic, which is a great way to get ideas about what might be true. Number theory has been a favorite of many famous mathematicians, and so some of their names are attached to the following important theorems.

3.1 Multiplicative inverses

Definition 3.1. An integer b is **divisible** by a non zero integer a, if there is an integer x such that $b = ax$. We write $a|b$. If b is not divisible by a, we write $a \not| b$.

Example 3.1. 14 is divisible by 7 because $14 = 7 \times 2$, and we write $7|14$.

Definition 3.2. The integer a is a **common divisor** of b and c if $a|b$ and $a|c$. Since there is a finite number of common divisors, the greatest one is called the **greatest common divisor** of b and c and is denoted by $\gcd(b, c)$.

Example 3.2. 6 is a common divisor of 24 and 120, but 24 is their greatest common divisor, i.e., $\gcd(24, 120) = 24$.

Definition 3.3. We say that a and b are **relatively prime** if $\gcd(a, b) = 1$.

Definition 3.4. An integer $p > 1$ is called a **prime number** or a **prime** if there is no divisor d of p satisfying $1 < d < p$. If an integer $a > 1$ is not a prime, it is a **composite number**.

Definition 3.5. The integer a is a **common multiple** of b and c if $b|a$ and $c|a$. The smallest common multiple of b and c is called the **least common multiple** and is denoted by $\text{lcm}(b, c)$.

Example 3.3. $(60)(84) = 5040$ is a common multiple of 60 and 84, but $(12)(7)(5) = 420$ is their least common multiple; $\text{lcm}(60, 84) = 420$.

Remark 3.1. Using prime factorizations (expressing an integer as a product of primes) it is easy to see that

$$\text{lcm}(b, c) = \frac{bc}{\gcd(b, c)}.$$

In our example with $b = 60$ and $c = 84$, we have $\gcd(60, 84) = 12$, so

$$\text{lcm}(60, 84) = (12)(7)(5) = \frac{(60)(84)}{12}.$$

Lemma 3.1. *Let a and n be integers with $0 < a < n$. Then a has a multiplicative inverse mod n if and only if row a of the residue multiplication table mod n is a permutation (rearrangement) of the residue numbers $0, 1, 2, \ldots n-1$. Furthermore, a does not have a multiplicative inverse mod n if and only if $az \equiv 0 \mod n$ for some $0 < z < n$.*

Proof. If a has a multiplicative inverse mod n, then both sides of the equation $ax \equiv ay$ mod n may be multiplied by a^{-1} to deduce $x \equiv y \mod n$.
Thus, if a^{-1} exists, then the residue entries of row a of the multiplication table are all distinct (different). Since there are n residue values and n entries in the row, we deduce that row a is a permutation of the n residue values.
Conversely, if row a is a permutation of the residue values, then the number "1" occurs somewhere in row a, say in column x. This means x is the multiplicative inverse of a. Thus we have shown that a^{-1} exists if and only if row a is a permutation of the residue values.
If a does not have a multiplicative inverse, then the number 1 does not appear in row a of the multiplication table. Since there are $n-1$ residue values besides 1, and n entries to fill, at least two of the entries of row a must be the same, say $ax \equiv ay$, with $0 \leq x < y < n$.
Thus $0 \equiv ay - ax \equiv a(y-x)$; i.e. the entry in column $z = y - x$ of row a is zero.
Conversely, if $az \equiv 0$ for some $0 < z < n$, then since column 0 and column z of row a in the table both have entries 0, row a is not a permutation of the residue numbers, so by the previous paragraph we deduce a^{-1} does not exist.

□

Theorem 3.1. *Let a and n be integers with $0 < a < n$. Then a has a multiplicative inverse mod n if and only if a and n are relatively prime, i.e.* $\gcd(a, n) = 1$.

Proof. We will check the logically equivalent statement that a does not have a multiplicative inverse if and only if $\gcd(a, n) = b > 1$.
If a does not have a multiplicative inverse then pick the smallest $0 < z < n$ so that $az \equiv 0$ mod n, which we can do by applying lemma 3.1.
Thus az is a multiple of n, and is in fact the least common multiple of a and n since by choosing the smallest positive z for which $az \equiv 0 \mod n$ we are choosing the smallest postive z so that az has n as a factor.
Since $z < n$ we also have $az < an$.
But $az = \text{lcm}(a, n) = \frac{an}{\gcd(a,n)}$, so it must be that $\gcd(a, n) > 1$.
Conversely, if $\gcd(a, n) = b > 1$, then for $z = \frac{n}{b}$ we have $az = \text{lcm}(a, n)$ so $az \equiv 0$ mod n, i.e. column z of row a of the multiplication table is zero, so a^{-1} does not exist by lemma 3.1. □

Remark 3.2. Although theorem 3.1 tells us when multiplicative inverses exist in clock arithmetic, it doesn't give us an efficient algorithm to compute them if the modulus is large. In example 3.4, we keep the modulus relatively small. In section **??**, we'll see how to find multiplicative inverses when the modulus is large.

Remark 3.3. Note that primes are special because all nonzero numbers mod p have a multiplicative inverse. We will use this fact in our public key algorithm.

Example 3.4. Find the multiplicative inverse of 8 mod 11.

Solution. We have already seen that we can find the multiplicative inverse by making a multiplication table, but we don't want to make that big of a table from scratch.
We could try to find the inverse by just going through the multiples of 8 until one of them is congruent to 1.
Here's a third way: we need a number b such that $8b \equiv 1 \mod 11$. The numbers congruent

to 1 mod 11 are 12, 23, 34, 45, 56, 67, 78, etc. Of those we need to find the one that is divisible by 8, which is $56 = 8 \times 7$. Thus the multiplicative inverse of 8 mod 11 is 7.

Exercise 3.1. Solve $8x \equiv 3 \mod 11$ for a residue number x.

3.2 Powers

Theorem 3.2 (Fermat's Little Theorem)**.** *If p is a prime and if $0 < a < p$ is a residue number, then $a^{p-1} \equiv 1 \mod p$.*

Proof. Pick any non-zero residue a as above, and consider the corresponding row of the mod p multiplication table. Since a has a multiplicative inverse mod p, $ax \equiv ay$ only when $x \equiv y$ (by Lemma 3.1). Therefore, as in our previous discussion of multiplication tables, the residues across the row, namely the residues of

$$1a, 2a, 3a, ..., (p-1)a$$

must all be different, i.e. a permuation of the non-zero residues $1, 2, ..., (p-1)$. Thus the product of all these terms satisfies

$$(1a)(2a)...(p-1)a \equiv (1)(2)...(p-1) \mod p,$$
$$a^{p-1}(1)(2)...(p-1) \equiv (1)(2)...(p-1) \mod p.$$

Multiply both sides of this equation by the multiplicative inverses of $2, 3, ...(p-1)$, i.e. cancel the term $(2)(3)...(p-1)$ from both sides of the equation. Deduce

$$a^{p-1} \equiv 1 \mod p.$$

□

Example 3.5. Here's how to illustrate Little Fermat concretely, using $p = 7$. Start with the mod 7 multiplication table, without the zero row and column:

mod 7 multiplication table

×	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

Take any row, say the row for $a = 3$. The entries going across are the residues for

$$(3)(1), (3)(2), (3)(3), (3)(4), (3)(5), (3)(6)$$

and they are just a permutation of the original non-zero residues. Thus, taking the product of the entries in this row, mod 7, we have

$$3^6 6! \equiv 6! \mod 7.$$

6! has a multiplicative inverse mod 7, since its a product of numbers with multiplicative inverses. Multiplying both sides of the equation by this number, we deduce a special case of Little Fermat, for $a = 3, p = 7$:

$$3^6 \equiv 1 \mod 7.$$

Theorem 3.3. *Let $N = p$ be a prime. Consider the power function $f(x) = x^e \mod p$ with domain equal to the residue numbers for p. Let e be chosen relatively prime to $p - 1$, $\gcd(e, p - 1) = 1$. Let d be a multiplicative inverse of e mod $p - 1$ $(0 < e, d < p - 1)$. Then $g(x) = x^d \mod p$ is the inverse function of f.*

Proof. Notice that if $x = 0$ the result holds. Thus we can assume $x = a$, a non-zero residue number for p.
Since e and d are multiplicative inverses mod $p - 1$, we have by Thm 3.1

$$ed = 1 + m(p - 1)$$

for some counting number m.
Thus

$$g(f(a)) \equiv g(a^e) \equiv (a^e)^d \equiv a^{ed}$$
$$\equiv a^{1+m(p-1)} \equiv a^1 (a^{p-1})^m \equiv a(1^m) \equiv a \mod p$$

by Fermat's Little Theorem. This shows that g is the inverse function to f. □

Example 3.6. For $p = 11$ and $e = 3$, find d using Thm 3.3. Does your answer agree with the earlier example, where we found acceptable d by brute force?

4 The Euclidean Algorithm and Multiplicative Inverses

The Euclidean Algorithm is a set of instructions for finding the greatest common divisor of any two positive integers. Its original importance was probably as a tool in construction and measurement; the algebraic problem of finding $\gcd(a, b)$ is equivalent to the following geometric measuring problem: Given two different rulers, say of lengths a and b, find a third ruler which is as long as possible, but so that you can still use it as a scale on both of the

longer rulers. The process, illustrated in Fig. 1, goes as follows. Since DC is shorter, it is used to measure BA. DC measures BA one time with remainder EA which is less than DC. Next EA is used to measure DC. EA measures DC twice with remainder FC shorter than EA. Finally, FC measures EA three times and there is no remainder. Looking backwards, we see that FC measures DC seven times and BA ten times.

The Euclidean Algorithm makes repeated used of integer division ideas. This procedure is called the division algorithm. If a and b are positive integers with $a < b$, then tarting with $(b)(0)$, then $(b)(1)$ etc. we may subtract increasing multiples of b from a until what remains is a non-negative number less than b. That is there exists numbers q and r with $0 \leq r < b$, such that

$$a = bq + r.$$

Here is the algebraic formulation of Euclid's Algorithm; it uses the division algorithm successively until $\gcd(a, b)$ pops out.

Theorem 4.1 (The Euclidean Algorithm)**.** *Given two integers $0 < b < a$, we make a repeated application of the division algorithm to obtain a series of division equations, which eventually terminate with a zero remainder:*

$$a = bq_1 + r_1, 0 < r_1 < b,$$

$$b = r_1q_2 + r_2, 0 < r_2 < r_1,$$

$$r_1 = r_2q_3 + r_3, 0 < r_3 < r_2,$$

$$\dots$$

$$r_{j-2} = r_{j-1}q_j + r_j, 0 < r_j < r_{j-1}$$

$$r_{j-1} = r_jq_{j+1}.$$

The greatest common divisor $\gcd(a, b)$ *of a and b is r_j, the last nonzero remainder in the division process.*

Let's look at an example of the Euclidean algorithm in action - it's really quick at finding gcd's when your two integers are large.

Example 4.1. Find the gcd of 42823 and 6409.

Solution. Step by steps division.

1. Divide (long division) 42823 by 6409:

$$\begin{array}{r} 6 \\ 6409\overline{)\,42823} \\ \underline{38454} \\ 4369 \end{array}$$

The quotient is 6 and the remainder 4369.

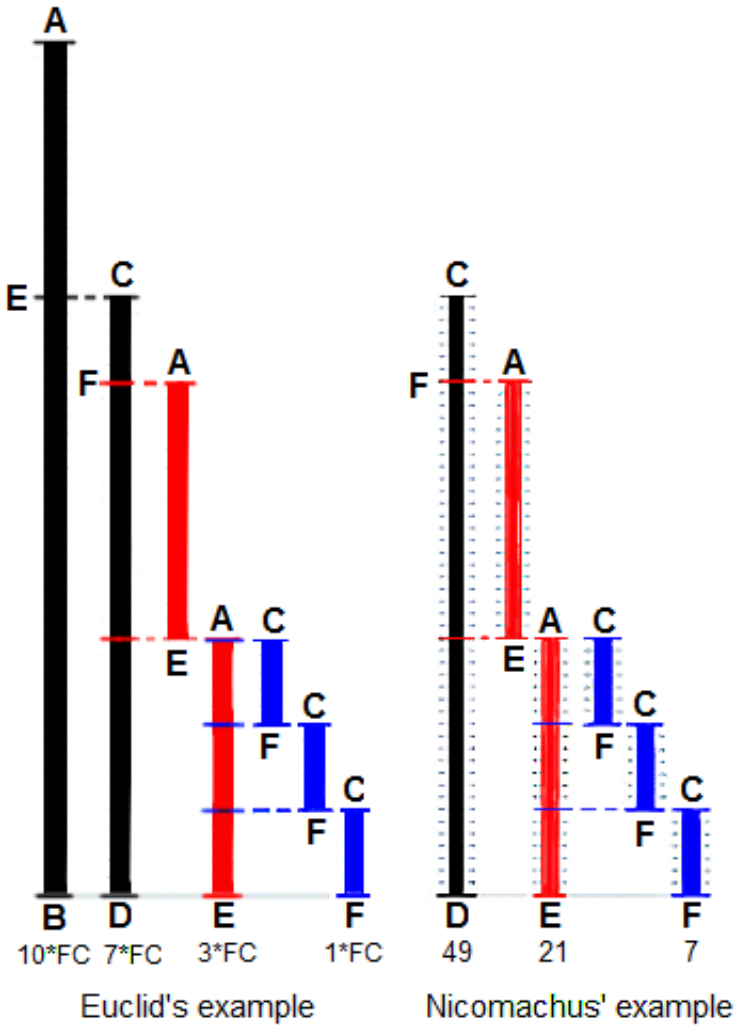


Figure 1: Euler's method for finding the gcd of two lengths BA and DC with gcd FC. Nicomachus' example with numbers 49 and 21 resulting in $\gcd(49, 21) = 7$ (from Wikipedia).

2. Divide (long division) 6409 by 4369:

$$\begin{array}{r} 1 \\ 4369\overline{)\,6409} \\ \underline{4369} \\ 2040 \end{array}$$

The quotient is 1 and the remainder is 2040.

3. Divide (long division) 4369 by 2040:

$$\begin{array}{r} 2 \\ 2040\overline{)\,4369} \\ \underline{4080} \\ 289 \end{array}$$

The quotient is 2 and the remainder is 289.

4. Divide (long division) 2040 by 289:

$$\begin{array}{r} 7 \\ 289\overline{)\,2040} \\ \underline{2023} \\ 17 \end{array}$$

The quotient is 17 and the remainder is 17.

5. Divide (long division) 289 by 17:

$$\begin{array}{r} 17 \\ 17\overline{)\,289} \\ \underline{170} \\ 119 \\ \underline{119} \\ 0 \end{array}$$

The quotient is 17 and the remainder is 0.

We can rewrite all the operations in one table, similarly as in the Euclidean Algorithm:

$$\begin{array}{rcrcr} \mathbf{42823} & = & \mathbf{6409}(6) & + & \mathbf{4369} \\ \mathbf{6409} & = & \mathbf{4369}(1) & + & \mathbf{2040} \\ \mathbf{4369} & = & \mathbf{2040}(2) & + & \mathbf{289} \\ \mathbf{2040} & = & \mathbf{289}(7) & + & \mathbf{17} \\ \mathbf{289} & = & \mathbf{17}(17) & & \end{array}$$

Therefore $\gcd(42823, 6409) = 17$.

Why does the Euclidean Algorithm actually give the gcd? It seems kind of strange that we can get the gcd of two numbers a and b by looking at the gcd's of the subsequent remainder values. Let's look at successive equations in this process.

Proof. From the first equation $a = bq_1 + r_1$, we deduce that since the gcd divides a and b it must divide r_1. Similarly in the next equation $b = r_1q_2 + r_2$, the gcd divides b and r_1,

so it must also divide r_2. Thus the $\gcd(a, b)$ divides all the remainders, including the final non-zero one, r_j. This implies that $\gcd(a, b) \leq r_j$.

On the other hand, by working up from the last equation $r_{j-1} = r_j q_{j+1}$ we deduce that r_j divides r_{j-1}. From the second to last equation r_j also divides r_{j-2}, and working all the way back to the top we see that r_j divides both a and b, so is a divisor. In other words, $r_j \geq \gcd(a, b)$.

Putting our reasoning together, r_j must be the greatest common divisor! □

Work on the following exercises in groups.

Exercise 4.1. Find the gcd's of the following pairs of numbers. Save your work because you'll need it later.

1. 7469 and 2464

2. 2689 and 4001

3. 2947 and 3997

4. 1109 and 4999

Euclid probably wasn't thinking about finding multiplicative inverses in modular arithmetic, but it turns out that if you look at his algorithm in reverse, that's exactly what it does ! The fact that we can use the Euclidean algorithm work in order to find multiplicative inverses follows from the following algorithm (back substitution).

Theorem 4.2 (Multiplicative Inverse Algorithm)**.** *Given two integers* $0 < b < a$*, consider the Euclidean Algorithm equations which yield* $\gcd(a, b) = r_j$*. Rewrite all of these equations except the last one, by solving for the remainders:*

$$r_1 = a - bq_1,$$

$$r_2 = b - r_1 q_2,$$

$$r_3 = r_1 - r_2 q_3,$$

$$\dots$$

$$r_{j-1} = r_{j-3} - r_{j-2} q_{j-1}$$

$$r_j = r_{j-2} - r_{j-1} q_j.$$

Then, in the last of these equations, $r_j = r_{j-2} - r_{j-1} q_j$*, replace* r_{j-1} *with its expression in terms of* r_{j-3} *and* r_{j-2} *from the equation immediately above it. Continue this process successively, replacing* $r_{j-2}, r_{j-3}, \dots$*, until you obtain the final equation*

$$r_j = ax + by,$$

with x and y integers. In the special case that $\gcd(a,b)=1$*, the integer equation reads*

$$1 = ax + by.$$

Therefore we deduce

$$1 \equiv by \mod a$$

so that (the residue of) y is the multiplicative inverse of $b \mod a$.

Example 4.2. Find integers x and y to satisfy

$$42823x + 6409y = 17.$$

Solution. We begin by solving our previous equations for the remainders. We have:
$4369 = 42823 - 6409(6)$
$2040 = 6409 - 4369$
$289 = 4369 - 2040(2)$
$17 = 2040 - 289(7)$
Now we do the substitutions starting with that last equation and working backwards and combining like terms along the way:

$$\begin{aligned} 17 &= 2040 - 289(7) = 2040 - (4369 - 2040(2))(7) = 2040(15) - 4369(7) \\ &= (6409 - 4369)(15) - 4369(7) = 6409(15) - 4369(22) \\ &= 6409(15) - (42823 - 6409(6))(22) = 6409(147) - 42823(22) \end{aligned}$$

Therefore $x = -22, y = 147$. □

Example 4.3. Find the multiplicative inverse of $8 \mod 11$, using the Euclidean Algorithm.

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that $\gcd(8,11) = 1$ and therefore the inverse of $8 \mod 11$ exists by the theorem 3.1. Then we'll solve for the remainders in the right column, before backsolving:

$\mathbf{11}$	$=$	$\mathbf{8}(1)$	$+$	$\mathbf{3}$	3	$=$	11	$-$ $8(1)$
$\mathbf{8}$	$=$	$\mathbf{3}(2)$	$+$	$\mathbf{2}$	2	$=$	8	$-$ $3(2)$
$\mathbf{3}$	$=$	$\mathbf{2}(1)$	$+$	$\mathbf{1}$	1	$=$	3	$-$ $2(1)$
$\mathbf{2}$	$=$	$\mathbf{1}(2)$						

Now reverse the process using the equations on the right.

$$\begin{aligned} 1 &= 3 - 2(1) = 3 - (8 - 3(2))(1) = 3 - (8 - (3(2)) = 3(3) - 8 \\ &= (11 - 8(1))(3) - 8 = 11(3) - 8(4) = 11(3) + 8(-4) \end{aligned}$$

Therefore $1 \equiv 8(-4) \mod 11$, or if we prefer a residue value for the multiplicative inverse,

$$1 \equiv 8(7) \mod 11.$$

□

Be careful about the order of the numbers. We do not want to accidentally switch the bolded numbers with the non-bolded numbers!

Exercise 4.2. Find the greatest common divisor g of the numbers 1819 and 3587, and then find integers x and y to satisfy

$$1819x + 3587y = g$$

Exercise 4.3. Find the multiplicative inverses of the following:

1. $50 \mod 71$

2. 43 mod 64

Exercise 4.4. Using the information from the previous exercise, solve the following equation for x and check your answer.

$$50x \equiv 63 \mod 71.$$

Exercise 4.5. Solve $12345x \equiv 6 \mod 54321$. Hint: First find the gcd.

5 Public Key Cryptography and RSA

What we will do for RSA cryptography (and what has been done in cryptography for a long time before RSA) is to make packets consisting of lots of letters, and encrypt those. In RSA, we will use HUGE moduli $N = pq$, which are products of two different prime numbers, and break messages into number packets which are residue numbers of N. Then we'll encrypt each packet using a power function $\mod N$ which permutes the residue numbers , and hope to decrypt it with another power function $\mod N$.

We can use the Little Fermat Theorem to understand power encryption/decryption when the modulus is a product of two different primes, and this is the basis of RSA cryptography.

Theorem 5.1. *(RSA decryption, when $N = pq$) Let $N = pq$ be a product of two distinct prime numbers. Define $N_2 := (p-1)(q-1)$. Let e be relatively prime to N_2. Then $f(x) = x^e \mod N$ has inverse function $g(x) = x^d \mod N$, where the domain and range for f and g are the residue numbers for N, and where d is the multiplicative inverse of e, mod N_2.*

Proof. If the hypotheses of the Theorem hold, then the claimed encryption and decryption powers e, d are related by

$$ed = 1 + m(p-1)(q-1)$$

for some counting number m. If we can show that

$$x^{ed} \equiv x \mod N$$

for all of $N's$ residue numbers, then it will follow that the modular power functions $f(x), g(x)$ are inverses of each other.
The trick is to show $x^{ed} - x$ is a multiple of p and also a multiple of q. Since p and q are different primes, the prime factorization of $x^{ed} - x$ must then include a factor of $pq = N$, so that $x^{ed} - x \equiv 0 \mod N$ as desired.
We show $x^{ed} - x$ is a multiple of p. If x is a multiple of p then this is automatic. Otherwise, $\gcd(x, p) = 1$, and the residue of $x \mod p$ is a non-zero number a. By Little Fermat,

$$x^{p-1} \equiv a^{p-1} \equiv 1 \mod p.$$

Thus

$$x^{ed} = x^{1+m(p-1)(q-1)} = x^1(x^{p-1})^{m(q-1)} \equiv x(1)^{m(q-1)} \equiv x \mod p.$$

Thus in both cases, $x^{ed} - x$ is a multiple of p. By repeating the argument above and interchanging the roles of p and q, we deduce that $x^{ed} - x$ is also a multiple of q. Thus $x^{ed} - x$ is a multiple of pq, since p and q are prime and have no common factors. In other words, $x^{ed} \equiv x \mod N$ as claimed. □

Example 5.1. For $p = 3$ and $q = 5$ we have $N = 15$, $N_2 = 8$. For the encryption power $e = 3$, compare the decryption power d guaranteed by this corollary to the power(s) we found earlier from the mod 15 power table.

Exercise 5.1. Let $p = 23, q = 41$, so that $N = 943$. Pick the encryption power $e = 7$. Find the auxiliary modulus N_2 and a decryption power d.

5.1 Alice and Bob

First, before exchanging encrypted messages, Alice and Bob do some preliminary work.

Alice Ⓐ

- Picks two large primes: p_A and q_A.
- Computes modulus $N_A = p_A q_A$.
- Picks encryption power e_A such that

$$\gcd(e_A, (p_A - 1)(q_A - 1)) = 1.$$

Public key: N_A | e_A

Bob Ⓑ

- Picks two large primes: p_B and q_B.
- Computes modulus $N_B = p_B q_B$.
- Picks encryption power e_B such that

$$\gcd(e_B, (p_B - 1)(q_B - 1)) = 1.$$

Public key: N_B | e_B

Scenario 1. Bob wants to send a secret MESSAGE to Alice.

Alice Ⓐ	Bob Ⓑ
	1. Ⓑ transcribes MESSAGE into an integer x (or several blocks if MESSAGE is too long) 2. Ⓑ encrypts message using Alice's public key: $N_A \mid e_A$ $y = E_A(x) = x^{e_A} \mod N_A$
3. Ⓐ knows her number theory and p_A and q_A so she can find her decryption power d_A by solving the multiplicative inverse equation $e_A d_A \equiv 1 \mod (p_A - 1)(q_A - 1)$	
4. Ⓐ decrypts the message $x \equiv D_A\left(y\right) \equiv y^{d_A} \mod N_A.$	

In the above transactions, Alice has no way of being sure that the message came from Bob. Eve can send a message to Alice pretending to be Bob!

Scenario 2. Bob wants to send a secret MESSAGE to Alice with **a secure signature**.

Alice Ⓐ

Public key: $N_A \mid e_A$

Private key: d_A

with $e_A d_A \equiv 1 \mod (p_A - 1)(q_A - 1)$.

Signature: $s_A \equiv$ integer(s) $< N_A$ transcribing to e.g. "signed by Alice".

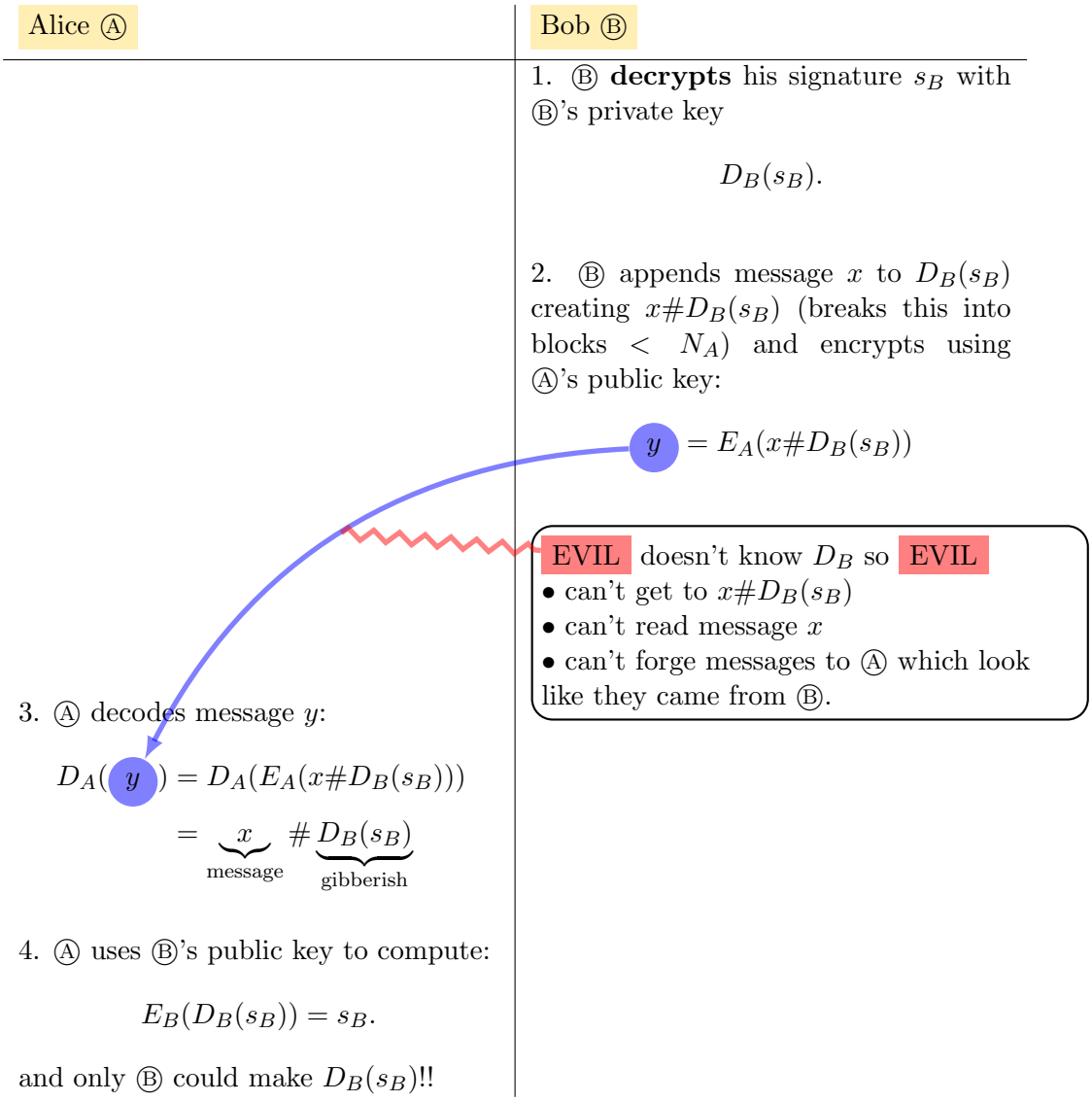
Bob Ⓑ

Public key: $N_B \mid e_B$

Private key: d_B

with $e_B d_B \equiv 1 \mod (p_B - 1)(q_B - 1)$.

Signature: $s_B \equiv$ integer(s) $< N_B$ transcribing to e.g. "signed by Bob".



Remark 5.1.

- There are quick and effective algorithms to check if a number is prime.
- In practice p and q are typically between 512 and 1024 bits long, i.e. those are the ranges for the number of digits in the binary representations. Converting to decimal, that means that p and q have between 155 and 308 base 10 digits, so that the modulus N has about twice as many.
- The public encryption function for residue numbers x is $E(x) = x^e \mod N$. This is a good public key system because it is believed that no one can figure out the decryption power d just knowing N, e. The reason this is believed to be true is that there is no known algorithm for factoring large composite numbers which is fast enough to recover p and q from sufficiently large N, at least in any reasonable time period like the age of the universe. There are slow algorithms: you could test every integer less than or equal to $\sqrt{(N)}$ as possible factors (since if N factors into two integers, the smaller one

will be at most this large). But if N is at least 10^{310} this requires on the order 10^{155} checks to find the smaller factor. Since the age of the universe is currently thought to be about $4(10^{17})$ seconds, this would require at least 10^{137} checks per second which is orders of magnitude beyond the capabilities of all the computers on earth working together.

- There are more efficient factorization algorithms, but none that are (publicly) known that could do this sort of factorization in any reasonable time frame. According to Wikipedia, the largest RSA modulus N with secret p, q that has been successfully factored in the public world was about 10^{231}.