INTRODUCTION TO NUMBER THEORY AND CRYPTOGRAPHY

IRENE RYU

Abstract. This paper introduces the basic idea behind cryptosystems and how number theory can be applied in constructing them. We begin with ciphers which do not require any math other than basic arithmetics. By exploring several such examples, we find a motivation for making use of number theory in constructing more efficient, complex cryptosystems.

Contents

1. Definitions

The following are recurring ideas used in the construction of cryptosystems.

Definition 1.1. If a number is divisible only by 1 and itself, we call this number a prime.

This notion is of great importance. Most generally, all integers can be represented as a product of primes. Also consider how primes are "rare" compared to composite values. Later on we will see how this rareness allows us to make locks with keys that are hard to find.

Definition 1.2. A number base is the number of different digits or combination of digits that a system uses to represent a number.

Date: August 15, 2020.

The decimal system is most prevalent, but the mechanics are identical regardless of the choice of base value. For instance,

$$\begin{aligned} 153 &= 15 \cdot 10 + 3 \\ 15 &= 1 \cdot 10 + 5 \\ 1 &= 0 \cdot 10 + 1 \\ 153 &= \big((0 \cdot 10 + 1) \cdot 10 + 5\big) \cdot 10 + 3 \\ &= 3 + 5 \cdot 10^1 + 1 \cdot 10^2 \end{aligned}$$

in base 10. But if we wanted to write 153 in base 2, we can go through a similar procedure to get 10011001_2.

$$2^0 \cdot 1 + 2^1 \cdot 0 + 2^2 \cdot 0 + 2^3 \cdot 1 + 2^4 \cdot 1 + 2^5 \cdot 0 + 2^6 \cdot 0 + 2^7 \cdot 1 = 153$$

This paper assumes some familiarity with modular arithmetic. If this is not the case, the most relatable instance of modular arithmetic can be found in how time is read (time in am/pm represents integers less than 24 evaluated modulo 12).

Definition 1.3. If a is congruent to b, or $a \equiv b \pmod{p}$, then $a - b = kp$ for some integer k.

Remark 1.4. If a and b are congruent modulo p, a and b will have the same remainder when divided by p. That is, a and b will belong to the same residue class. The least residue of a mod p is an integer r such that $r \equiv a \bmod p$ and $0 \leq r < p$. It is easy to check that there is only one such value. Addition, subtraction and multiplication between residue classes are similar to those in real numbers.

Definition 1.5. Two values are considered coprime if their greatest common divisor is 1.

Remark 1.6. The notion of coprime is needed to define division in modular arithmetic. Be warned, division of residue classes is very different from that in real numbers. In order to "divide", the number you are dividing *by* must be invertible. In $\mathbb{R}$, it is convenient to denote the inverse of some value k as $1/k$, and luckily there is a corresponding real value that can be assigned to this inverse for every $k \in \mathbb{R} \setminus \{0\}$. This is not the case in all rings. Recall that an element k in a ring has an inverse, or is invertible, if there exists some other element in that ring, denoted by k^{-1}, such that $k \cdot k^{-1} = 1$. As we are concerned with modular arithmetic, we can let division be defined simply as the multiplication by an inverse.

Proposition 1.7. *An integer a is invertible mod m if and only if a is coprime to m.*

Proof. If a is invertible modulo m, then there exists b such that $a \cdot b \equiv 1 \pmod{m}$. By definition of congruence, this means that there exists some integer k such that $ab - 1 = mk$. Suppose the gcd of a and m is some value d. Then by definition of gcd, d divides both a and m. Then going back to our equation

$$ab - 1 = mk.$$

Since $d|mk$, the left hand side must also be divisible by d. We already know that $d|ab$, so we must have that $d|1$. Thus $d = 1$.

For the converse, the statement that a and m are coprime is equivalent to stating $gcd(a, m) = 1$. Then, as we will see from the Euclidean Algorithm, there exist some x, y such that

$$ax + my = 1.$$

Since $my \equiv 0 \pmod{m}$, we have that $ax \equiv 1 \bmod m$, so a is invertible modulo m. □

Let us introduce the Euclidean Algorithm. Suppose we have two integers a, b such that $0 < b < a$. Repeat the following division algorithm until the remainder is zero.

$$\begin{aligned} a &= q_1 b + r_1 \\ b &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{k-1} &= q_k r_k + r_{k+1} \\ r_k &= q_{k+1} r_{k+1} \end{aligned}$$

The last nonzero remainder is the $gcd(a, b)$. Since we are consecutively dividing the left hand side by the remainder from the previous equation, we have that $b > r_1 > \ldots > r_k$. Notice the values are strictly decreasing and nonnegative, so the algorithm must terminate. Now we will show that r_{k+1} is in fact equal to $gcd(a, b)$.

Lemma 1.8. *The greatest common divisor of a and b is equal to r_{k+1}.*

Proof. We want to show that $gcd(a, b) = gcd(r_{i-1}, r_i)$, for all $0 < i \leq k + 1$ where $r_0 = b$. Let us consider the first equation (where $i = 1$). Define the set $D = \{d : d|a \text{ and } d|b\}$ where $d' = max\{D\} = gcd(a, b)$. Then

$$\begin{aligned} a - q_1 b &= r_1 \\ d'x &= r_1 \quad \text{for some } x \in \mathbb{Z}. \end{aligned}$$

So for all d, $d|r_1$. We want to show $gcd(b, r_1)$ is exactly equal to d', and not a multiple of d'. $\frac{q_1 b + r_1}{d'}$ is coprime to $\frac{b}{d'}$. That is,

$$q_1(\frac{b}{d'}) + \frac{r_1}{d'} \text{ is coprime to } \frac{b}{d'}.$$

If $gcd(b, r_1) = kd'$ for some integer k, then $\frac{r_1}{d'}$ and $\frac{b}{d'}$ are not coprime. This contradicts our assumption that $q_1(\frac{b}{d'}) + \frac{r_1}{d'}$is coprime to$\frac{b}{d'}$.

Now recall that we must arrive at an equation with no remainder by our claim that the algorithm will ultimately terminate. This will give us a final equation of the form

$$r_k = q_{k+1} r_{k+1}.$$

Since $gcd(a, b) = gcd(r_{i-1}, r_i)$, for all $0 < i \leq k + 1$, it follows that $gcd(a, b) = gcd(r_k, r_{k+1}) = r_{k+1}$. □

We have just proved the following result:

Proposition 1.9. *The Euclidean Algorithm provides an efficient method to calculate the gcd of two values a and b. Moreover, it can be used to represent d as a linear combination of a and b.*

You may wonder what advantages the Euclidean Algorithm has over prime factorization: for larger numbers it is difficult to guess some common divisor to reduce the two values. This is easy in the case of three or four digit values but we will be concerned with larger integers.

Example 1.10. Finding the gcd of 30 and 42 using the Euclidean Algorithm:

$$42 = 1 \cdot 30 + 12$$
$$30 = 2 \cdot 12 + 6$$
$$12 = 2 \cdot 6$$

Now, if we want to express 42 in terms of the partial remainders, we start with the first equation and successively plug the following equations one by one. In our case this gives us

$$42 = 1 \cdot \big[2(2 \cdot 6) + 6\big] + 12.$$

Rearranging, the Euclidean Algorithm gives us the $gcd(42, 30)$ as a linear combination of 42 and 30:

$$6 = 3 \cdot 30 - 2 \cdot 42.$$

2. "Simple" Ciphers

If you want to encode a secret message, you create a cipher. This does not necessarily call for the use of number theory, but I hope to convince you why using number theory might be a good idea. Conventionally, the alphabet corresponds to integer values 0 to 25 in order from a to z. We will call the original message the plaintext.

Let us first consider one to one substitutions. We assign a new value to correspond to each letter of the plaintext. The choice can be random or follow some formal procedure, but it is easy to see the two will not make much of a difference as far as it is a simple one to one substitution. But in order to see how the structure of ciphers can get complicated, let us introduce two formal methods of substitution, namely a caesar shift and and an affine cipher.

Example 2.1. Shift Cipher
If we take x to be an integer value of our plaintext, we define a function $f(x) = ax+b \pmod{26}$ to send it to the encoded value. Clearly this procedure will be one to one when $a = 1$ since the function will simply shift all integers by b; we call this a caesar shift. Consider now the case of an affine cipher, meaning that $a \neq 1$. For the function to be one to one, a must be chosen to be a unit modulo 26. We can decode the cipher with the inverse function $f^{-1}(x) = a^{-1}(x-b)$ if and only if there exists an inverse of a modulo 26.

The most notable shortcoming of one to one substitutions is that it is vulnerable to frequency analysis. Since the function is one to one, if the enemy figures out the decrypted value of one letter, they can rely on the fact that this value will always decrypt to the same letter. Then, the frequency of the letters will be consistent

with that of the original text which allows the enemy to break the code with relative ease.

Luckily, we can introduce methods to strengthen simple substitution ciphers. These fall into the category of polyalphabetic ciphers. Polyalphabetic ciphers allow different encodings of the same letter, depending on the key or the location of the letter within the plaintext.

Example 2.2. Block Cipher
In a simple shift cipher, all values were encoded under the same function given by $f(x) = ax + b$. In a block cipher, we encrypt multiple letters in a single block, but the "function" may vary within a single block for each place value.

Let us consider the **Vigenere cipher** as a general example. Suppose we want to encode the plaintext *happy house*.

(1) Choose a keyword of length n: let us choose *purple* to be our keyword.
(2) Define a vector of length n, which in our case would be the following:

keyword	p	u	r	p	l	e
number	15	20	17	15	11	4

(3) Break up our plaintext into blocks of length n and add the vector defined by our keyword

message	h	a	p	p	y	h	o	u	s	e
number	7	0	15	15	24	7	14	20	18	4
keyword	p	u	r	p	l	e	p	u	r	p
keyword number	15	20	17	15	11	4	15	20	17	15
encoding	22	20	6	4	9	11	3	14	9	19
ciphertext	W	U	G	E	J	L	D	O	J	T

Notice that the letter "p" in the message is encoded to different values in the ciphertext depending on the vector component added to it. The same is true for the letter "h". Although a block cipher is more resilient to frequency attacks, if the length of the key is determined, one can simply break apart the ciphertext into n groups, after which determining the shift for each group will be equivalent to finding a key of length one.

Example 2.3. Transposition Cipher
The **Playfair cipher** is a clever way of encoding a message without using any corresponding integer values. However, this example will demonstrate how regardless of the strategy, a non random encoding method of a ciphertext with components corresponding directly to the components of the plaintext will be susceptible to brute force attacks, since the ciphertext itself reveals enough information to chip away towards the original message.

We will once again encode the message *happy house*, this time using the keyword *blueskyrain*.

(1) Construct a 5x5 encoding grid using the keyword (let i=j by convention).

b	l	u	e	s
k	y	r	a	ij
n	c	d	f	g
h	m	o	p	q
t	v	w	x	z

We have listed the remaining letters of the alphabet in order after the keyword. Notice the choice of a keyword with no duplicates, so there is no need to additionally eliminate.

(2) Break up the plaintext into digrams: *ha pp yh ou se*. If there are duplicates, add an x in between them and at the end to pair with the last letter: *ha px py ho us ex.*

(3) Encode by the following rules, remembering that we will be encoding digrams to digrams.
   - Find the first component of the digram on the grid. Then slide right and left until you locate the column with the second component of the digram. This is the first value of your encoded digram. For our example, we find the first component of *ha* to be *p*.
   - Now go back to the first component of the plaintext digram on the grid and slide up and down until you locate the row that holds the second component of the digram. This is the second value. Going back to our example, we find that *ha* encodes to *pk*.
   - If values are on the same row then shift right by one, wrapping around the grid if necessary.
   - If values are on the same column then shift down by one, wrapping around the grid if necessary.

   Following the given procedure, we obtain the encoded digram: *pk xs ma mp ek as.*

Although the Playfair cipher is a relatively convenient strategy, it is nevertheless vulnerable to similar attacks as previous ciphers. Specifically, since the Playfair cipher is simply a substitution encoding on digrams, we can use this hint to work towards a frequency analysis.

The key takeaway is that plaintexts encoded as ciphertexts by substitution have critical vulnerabilities (namely to frequency analysis). These weaknesses cannot be overcome by utilizing variations of the tools introduced thus far. Then, we turn to number theory hoping to find a new approach to our dilemma.

3. Fundamental Theorems in Number Theory

We must shortly digress from the topic of ciphers to establish some theorems which will make recurring appearances in our future work.

First, consider the following dilemma: you have a basket of eggs, and want to guess how many eggs are in this basket. You know that if you take out 3, 4 or 5 eggs at a time, you always have one left in the end. Additionally, if you take out 7 at a time, you don't have any left. Notice this is equivalent to solving the system of congruences given by

$$x \equiv 1 \pmod 3, \quad x \equiv 1 \pmod 4, \quad x \equiv 1 \pmod 5. \quad x \equiv 0 \pmod 7).$$

3.1. The Chinese Remainder Theorem.

Theorem 3.1. *Suppose we have pairwise coprime moduli denoted by* $m_1, m_2, \ldots, m_k \in \mathbb{N}$ *and arbitrary integers* $r_1, r_2, \ldots, r_k$ *for* $k \geq 2$*. Then there exists some least residue* a *mod* m*, where* $m = m_1 m_2 \ldots m_k$*, such that*

$$a \equiv r_j \ (mod\ m_j), \quad for\ 1 \leq j \leq k.$$

Proof. We can define $a_0 \equiv a \pmod{m}$ by taking

$$a_0 = \sum_{j=1}^{k} (m/m_j) c_j r_j,$$

where c_j is an integer value such that $(m/m_j)c_j \equiv 1 \pmod{m_j}$. We know such c_j exist because we have defined m_j for $1 \leq j \leq k$ to be pairwise coprime, so $\gcd(m/m_j, m_j) = 1$. Then we know (m/m_j) has an inverse mod m_j for all $1 \leq j \leq k$, which is denoted by c_j for our purpose. Now,

$$a = a_0 \equiv (m/m_j)c_j r_j \equiv r_j \pmod{m_j} \quad \text{for } 1 \leq j \leq k,$$

since for each term of a_0 such that $i \neq j$, $(m/m_j)c_j \equiv 0 \pmod{m_i}$. □

Now, going back to our basket of eggs, we can easily solve our system of congruences using this proposition.

- $m = 3 \cdot 4 \cdot 5 \cdot 7 = 420$.
- $m/m_1 = 420/3 = 140$,
  $m/m_2 = 420/4 = 105$,
  $m/m_3 = 420/5 = 84$.
  (we don't need to find m/m_4 since $r_4 = 0$, so the term will yield zero regardless)
- $c_1 = 2$ since $140 \cdot 2 \equiv 2 \cdot 2 \equiv 1 (\text{mod } 3)$. Similarly, we find $c_2 = 1$, $c_3 = 4$.
- $a_0 = 140 \cdot 2 + 105 \cdot 1 + 84 \cdot 4 = 721$.
- Finding the least residue, we get $721 \equiv 301 (\text{mod } 420)$.

Proposition 3.2. *If $d|m$ and $gcd(d, m/d) = 1$, then $x \equiv a \ (mod\ m)$ implies*

$$x \equiv a \ (mod\ d);$$
$$x \equiv a \ (mod\ m/d).$$

See Section 2.2.3 of [2] for a proof.

3.2. Euler's φ-Function and Euler's Theorem.

Definition 3.3. Let m be a positive integer. The set

$$Z_m := \{0, 1, \ldots, m-1\} \subset \mathbb{Z}$$

is called the least residue system modulo m.

Notice that every integer will be sorted into exactly one of these sets. In general, we do not have to choose the least residue to define a residue system. In fact, we can choose any m number of elements which are pairwise incongruent mod m and these will form a *complete residue system*.

Definition 3.4. For $m \in \mathbb{N}$, the number of elements in Z_m that are coprime to m is denoted by $\varphi(m)$. We call this function Euler's totient function.

Theorem 3.5. Euler's Theorem
Take $m \in \mathbb{N}$ and some integer a. If $gcd(a, m) = 1$, then

$$a^{\varphi(m)} = 1 (mod\ m).$$

Proof. Consider R_m, the subset of Z_m consisting of all the elements coprime to m. So:

$$R_m = \{r_1, \ldots, r_{\varphi(m)}\}.$$

Notice that the elements of R_m will remain incongruent if we multiply all of them by some integer a such that $gcd(a, m) = 1$. Moreover, since we chose a to be coprime with m, the resulting elements $ar_1, \ldots, ar_{\varphi(m)}$ represent the elements of R_m written in a different order. Therefore,

$$(ar_1) \cdots (ar_{\varphi(m)}) \equiv r_1 \cdots r_{\varphi(m)} \pmod{m}.$$

Then multiplying each side with the inverse of $r_1 \cdots r_{\varphi(m)} \pmod{m}$, we obtain our result. □

Corollary 3.6. Fermat's Little Theorem
Let p be a prime and let a be an integer such that $p \nmid a$. then $a^{p-1} \equiv 1 (mod\ p)$.

Proof. First notice for any prime number p, $\varphi(p) = p - 1$. Combining this result with Euler's Theorem, we have that for any a not divisible by p,

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

□

3.3. **Orders and Squaring.**

Definition 3.7. If u is a unit modulo m, the smallest $k > 0$ such that $u^k \equiv 1 \pmod{m}$ is called the order of u.

In order to compute $a^k \pmod{m}$ more efficiently, we introduce the squaring algorithm.

Definition 3.8. Successive Squaring

- Find the binary expansion of $k = b_j b_{j-1} \ldots b_0$ (we will be needing this at the final step).
- Compute $a^2, a^4, \cdots, a^{2^d}$ by squaring the previous entry and reducing mod m at each step.
- Using the binary expansion obtained at the first step, we have

$$a^k = \prod_{0 \leq i \leq j} a^{b_i} \pmod{m}.$$

Example 3.9. Compute $2^{516} \pmod{61}$

- $516 = 1000000100_2$, meaning $516 = 2^2 + 2^9$. So $2^{516} = 2^{512} \cdot 2^4$
- By successively squaring and reducing, we obtain all values of 2^{2^d}. But ultimately we only use $2^{512} \equiv -4 \pmod{61}$ and $2^4 \equiv 16 \pmod{61}$.
- $2^{516} = 2^{512} \cdot 2^4 \equiv (-4) \cdot 16 \equiv -64 \equiv 58 \pmod{61}$

There exist variations of the squaring algorithm, but the underlying mechanisms are similar.

4. Public-Key Cryptosystems

Going back to our discussion of cyphers and cryptosystems, we now introduce the notion of a public key. Suppose Alice wants to send Bob a message, and Eve can intercept the encrypted message. Bob establishes a *public key* which is shared with everyone, and this allows anyone to send him an encoded message. However, the key is constructed in a way such that only Bob will be able to decrypt the message. That is, a good public key must not only be relatively simple to decode, but it also must be secure so that even if Eve intercepts Alice's message, she will not be able to decode it.

4.1. **Rabin Encryption.** The Rabin Encryption uses the idea that it is difficult to factor out large prime numbers.

- Bob chooses two large prime numbers p and q. The product $N = pq$ is his key. N will be public, but only Bob knows the values p and q.
- If Alice wants to send Bob a message, she converts her message into an integer m modulo N. Then she computes m^2 modulo N, and sends this to Bob.
- Since Bob knows p and q, using Proposition 3.2, he can solve the system of congruences given by

$$x^2 \equiv a \pmod{p}, \quad x^2 \equiv a \pmod{q}, \quad \text{where } a = m^2.$$

- Notice, if $x^2 \equiv m^2 \pmod{p}$ then $p|(x-m)(x+m)$. Thus $x \equiv \pm m \pmod{p}$. Therefore if Bob finds one solution, he immediately gets the other for both congruences.
- Bob can easily compute x using $x = a^{(p+1)/4}$.
  We have that $a = m^2 \pmod{p}$ and $m^{p-1} \equiv 1 (\text{mod } p)$ by Theorem 3.5. We want to derive an x such that $x^2 \equiv a \pmod{p}$:

$$a \equiv m^2 \equiv m^{p-1}m^2 = m^{p+1} \equiv a^{(p+1)}/2 \equiv x^2 (\text{mod } p).$$

- Using Theorem 3.1, Bob can calculates the congruences

$$x \equiv \pm a^{(p+1)/4} (\text{mod } p)$$
$$x \equiv \pm a^{(q+1)/4} (\text{mod } q)$$

  to obtain four possible values.

Luckily, there is an improved version of the Rabin Encryption which eliminates the nonuniqueness of roots.

4.2. **RSA Encryption.** The RSA encryption goes through the same procedure as the Rabin Encryption, except it does not use a one-to-one map. This allows the RSA encryption to resolve the issue of multiple roots in the Rabin Encryption. In an RSA Encryption, Bob chooses some power e so that both N and e serve as his public key, instead of defaulting to a squaring map. Moreover, Bob must choose an integer e that is relatively prime to $\varphi(N) = (p-1)(q-1)$; the idea is that he can easily decode Alice's message $c \equiv m^e$ modulo N by computing c^d modulo N, where d is the inverse of e modulo $\varphi(N)$.

Theorem 4.1. *If $de \equiv 1 \pmod{\varphi(N)}$ and $c \equiv m^e \pmod{N}$, then $c^d \equiv m \pmod{N}$.*

Proof. We will use the Chinese Remainder Theorem (Theorem 3.1) and Fermat's Little Theorem (Theorem 3.6). First, our assumption implies that $c^d \equiv m \pmod{p}$ and $c^d \equiv m \pmod{q}$, since $N = pq$. Since $\varphi(N) = (p-1)(q-1)$, if $de \equiv 1 \pmod{\varphi(N)}$ then $de \equiv 1 \pmod{p-1}$. So

$$de = 1 + k(p-1),$$

for some $k \in \mathbb{Z}$. Since $c \equiv m^e \pmod{p}$,

$$c^d \equiv m^{de} \equiv m^{1+k(p-1)} \equiv m \cdot (m^{p-1})^k \pmod{p}.$$

Now, if $m \equiv 0 \pmod{p}$, then $c^d \equiv 0 \equiv m \pmod{p}$ so the result holds. If $p \nmid$ m then we use Fermat's little theorem, which gives us $c^d \equiv m \cdot 1^k \equiv m \pmod{p}$. Thus, our assumption holds. □

As an additional note, using the squaring algorithms previously introduced we can calculate the value of c^d quite efficiently.

4.3. **Pollard's *p-1* Algorithm.** The public key encryption systems introduced in the previous section are an upgrade from the basic cryptosystems in **Section 2**. But this final section is dedicated to the skeptics who will argue (rightfully so), "how difficult would it be to factor out a product of two large prime numbers?".

To begin with, by the Prime Number Theorem the approximate number of primes less than X is $\frac{X}{lnX}$. The probability of some large integer N being a prime is about $\frac{1}{lnN}$; going back to the case of an RSA encryption, Bob will have a more secure key by choosing a larger p and q, but this would also make the decoding process more difficult. Thus, Bob must find a balance in choosing his public key. There are ways to determine whether an integer is composite or prime, but it is generally easier to determine if a value is composite than to determine whether it is a prime or not.

As a response to the critics of the RSA encryption system, let us explore a method to determine the divisors for $N = pq$.

We start from the assumption that for any residue a modulo n, the order of a modulo p and a modulo q will not be the same. Let k be the order of a modulo p and suppose that it is smaller than the order of a modulo q. Then

$$a^k \equiv 1 \pmod{p}, \quad a^k \not\equiv 1 \pmod{q};$$

so $gcd(a^k - 1, n) = p$. Notice that for any multiple of the order k, our conditions will still hold; this will make the computation much simpler. We just have to make sure that kx (for some $x \in \mathbb{Z}$) is not also a multiple of the order of a modulo q since that would make the greatest common divisor n.

Then, if p is a prime divisor of n such that $p-1$ only has small prime factors, the order of any element modulo p will have frequent multiples. Using this idea, we can find a way to quickly calculate possible values that will give us $gcd(a^k - 1, n) = p$.

More explicitly, we want to set a bound B and evaluate the values $a^{1!}, a^{2!}, \ldots, a^{B!}$ to ultimately compute $\gcd(a^{B!} - 1, n)$.

If the result is 1, then you can either find a new residue a and start over, or continue on with calculations by raising $a^{B!}$ to consecutive powers. If we get a value between 1 and n, we are done as we have found p. Finally, if the gcd is n, then the bound B is too large, so you must go back and repeat the process with a smaller bound. If the algorithm finds an exponent x such that $gcd(a^x - 1, n) = p$, then we can definitely say that n is composite. However, not finding such an element does not tell us anything about the nature of n.

5. Concluding Remarks

Hopefully this paper has convinced you of the infinite possibilities in the field of cryptosystems. The purpose of this expository paper was not so much focused on the specific examples or theorems, but rather on intriguing the reader, as I was, on finding out more. Clearly there does not exist an unbreakeable code (apart from a one-time pad, which has its own shortcomings), but we can continue to construct new tools and methods in the hope for better models.

Acknowledgments

I would like to thank my mentor Andreea Iorga for helping me navigate my way through cryptosystems. I would have been very very lost without her help. I would also like to thank Professor May for organizing this program and for accepting my late application. Finally, thank you Daniil for buying a glowing blackboard and taking the time and effort to provide us with good content in a remote environment.

References

[1] Evan Dummit. Cryptography (part 1) Classical Cryptosystems and Modular Arithmetics. https://web.northeastern.edu/dummit/docs/cryptography_1_classical_cryptosystems.pdf

[2] Evan Dummit. Cryptography (part 2) Public-Key Cryptography. https://web.northeastern.edu/dummit/docs/cryptography_2_public_key_cryptography.pdf

[3] Harald Niederreiter, Arne Winterhof. Applied Number Theory. Springer 2015