NUMBER THEORY AND CRYPTOGRAPHY

KEITH CONRAD

1. Introduction

Cryptography is the study of secret messages. For most of human history, cryptography was important primarily for military or diplomatic purposes (look up the Zimmermann telegram for an instance where these two themes collided), but internet commerce in the late 20th century made cryptography important for everyone. Anytime you send a text message, buy something online, or transfer funds electronically, cryptography is at work to guarantee – or so you hope – that nobody sees your information other than the intended recipient. The most basic cryptographic protocols, used today by millions worldwide without realizing it, are based on number theory. In Section 2 we will discuss some cryptographic techniques used before the computer era that involve modular arithmetic and linear algebra. In Sections 3-5 we will describe one of the most widely used cryptographic protocols today, called RSA after its inventors Rivest, Shamir, and Adleman [8]. It is based on exponentiation in modular arithmetic, and the math behind it is Euclid's algorithm, Fermat's little theorem, and primality testing. Section 6 discusses some of the history behind who found RSA.

In addition to cryptography, which aims to keep messages secret from eavesdroppers, there is a related area called coding theory. The goal in coding theory is not to hide messages, but to make sure they pass through a noisy channel without errors. This is important in telephone networks, digital music files, and communication between NASA on Earth and satellites in space. Number theory plays a role in coding theory, but it is not what we will be discussing here.

2. The Caeser and Hill Ciphers

One of the oldest methods of encryption, which goes back to Julius Caesar, shifts every letter in a message by a fixed amount. For instance, if we shift by 3 letters to the right, then we have

$$\mathrm{A} \mapsto \mathrm{D}, \mathrm{B} \mapsto \mathrm{E}, \mathrm{C} \mapsto \mathrm{F}, \ldots, \mathrm{Z} \mapsto \mathrm{C}.$$

The message

DOODLE

is encrypted as

GRRGOH.

The encrypted message

XKFJXI

is decrypted by shifting by three letters to the left, giving us

ANIMAL.

This process of encrypting by shifting a fixed amount is called a *Caesar cipher* or a *shift cipher*. It has (at least) three defects:

(1) Knowing the encrypting process reveals the decrypting process: if we encrypt by shifting to the right by five letters then decrypting is just shifting to the left by five letters. Therefore it is important to keep the shift value secret to make the method secure, although the next defect shows the whole idea behind the Caesar cipher is sort of doomed.
(2) Even if we don't know the exact shift amount, the search space is very small: each letter has only 25 shifts besides itself, so if we know a message was encrypted by some Caesar cipher then a little patience will reveal the message by just trying all the possibilities until you shift into a text that makes sense.
(3) Because every letter is shifted by the same amount, once we know the encrypted form of one letter we know the shift for all letters, *e.g.*, if we determine that B is encrypted as F then B is shifted by 4 letters so every letter is shifted by 4 letters.

Mathematically, the Caesar cipher is addition by a fixed number modulo 26, where we take A = 1, B = 2, and so on up to Z = 26 = 0. The shift by 3 letters to the right is the encryption function $E(x) = x + 3 \bmod 26$. It is decrypted with $D(y) = y - 3 \bmod 26$. We can get something a bit more complicated by allowing scaling before translating: $E(x) = ax + b \bmod 26$, where $(a, 26) = 1$. A Caesar cipher uses $a = 1$. We need $(a, 26) = 1$ in order to invert $E(x)$, *i.e.*, to decrypt a message. Suppose $E(x) = 5x + 3 \bmod 26$, so letting $x = 1, 2, 3, \dots, 25, 0$ gives the encryption

$$\mathrm{A} \mapsto \mathrm{H}, \mathrm{B} \mapsto \mathrm{M}, \dots, \mathrm{Z} \mapsto \mathrm{C}.$$

When $E(x) = x + 3 \bmod 26$ we found that DOODLE becomes GRRGOH. If $E(x) = 5x + 3 \bmod 26$ then DOODLE becomes WZZWKB (check!) and if $E(x) = 9x + 3 \bmod 26$ then DOODLE becomes MHHMGV (check!). Decryption is based on inverting a linear function. In ordinary algebra, if $y = 5x + 3$ then $x = (1/5)(y - 3)$. This works modulo 26 as well, replacing 1/5 with the inverse of 5 mod 26, which is 21 (check!). Therefore if $E(x) = 5x + 3 \bmod 26$ then the corresponding decryption function is $D(y) = 21(y - 3) = 21y + 15$ (because $21(-3) = -63 \equiv 15 \bmod 26$).

By allowing a scaling mod 26 as part of the encryption, we enlarge the number of encryption functions by more than a factor of 10: the number of encryption functions of the form $E(x) = x + b \bmod 26$ is 26 (really it's 25, because the encryption function $E(x) = x \bmod 26$ is stupid), while the number of encryption functions of the form $E(x) = ax + b \bmod 26$ is $\varphi(26)26 = 12 \cdot 26 = 312$.

Even though this generalization of the Caesar cipher avoids the problem of every letter being shifted by the same amount, it shares a flaw with the Caesar cipher that we did not bring up earlier: a letter gets encrypted in the same way no matter where it appears in the text. For example, when DOODLE is encrypted as GRRGOH with $E(x) = x + 3 \bmod 26$, the D became G both times and the double O becomes a double R. When DOODLE turns into WZZWKB using $E(x) = 5x + 3 \bmod 26$, the D becomes W both times and the double O becomes a double Z. The frequency statistics of letters in a language are well-known, *e.g.*, the letter E is the most frequently appearing letter in plain English text and the second most common letter is A. Therefore if we have a long amount of encrypted text and we know it was encrypted by some function $E(x) = ax + b \bmod 26$, we could determine with reasonable confidence which encrypted letters correspond to E and A. Just as two points determine a line, knowing two values of the function $E(x) = ax + b \bmod 26$ is enough to let us recover a and b, which tells us the encryption function.

We can get a more sophisticated extension of the Caesar cipher by working with vectors and matrices instead of numbers and linear functions. This is called a *block cipher* because it operates on groups of letters at a time instead of one letter at a time. We will describe the idea using two-letter blocks with 2×2 matrices. For instance,

DOODLE

becomes

DO OD LE

and each block can be viewed as a vector of numbers mod 26 (A = 1, B = 2, and so on):

$$\begin{pmatrix} \mathrm{D} \\ \mathrm{O} \end{pmatrix} = \begin{pmatrix} 4 \\ 15 \end{pmatrix}, \quad \begin{pmatrix} \mathrm{O} \\ \mathrm{D} \end{pmatrix} = \begin{pmatrix} 15 \\ 4 \end{pmatrix}, \quad \begin{pmatrix} \mathrm{L} \\ \mathrm{E} \end{pmatrix} = \begin{pmatrix} 12 \\ 5 \end{pmatrix}.$$

Pick a 2×2 matrix with entries mod 26, such as

$$\mathcal{A} = \begin{pmatrix} 3 & 2 \\ 1 & 11 \end{pmatrix}.$$

To encrypt a block of two letters, viewed as a vector with two components mod 26, multiply the vector by the matrix $\mathcal{A}$ and do the calculations mod 26:

$$\mathcal{A}\begin{pmatrix} 4 \\ 15 \end{pmatrix} = \begin{pmatrix} 16 \\ 13 \end{pmatrix}, \quad \mathcal{A}\begin{pmatrix} 15 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \\ 7 \end{pmatrix}, \quad \mathcal{A}\begin{pmatrix} 12 \\ 5 \end{pmatrix} = \begin{pmatrix} 20 \\ 15 \end{pmatrix}.$$

Turning each vector component back into a letter, the encrypted form of DOODLE has first letter 16 = P, second letter 13 = M, third letter 1 = A, and so on, giving us

PMAGTO.

Notice that the D's in DOODLE have been encrypted by different letters, and likewise for the O's. This system won't be cracked by knowledge of the frequency of single letters in English, although it would be susceptible to analysis based on a frequency of double letter combinations in English.

Let's turn to decryption. Suppose we received the message

UPFOOU

and we know it was encrypted by multiplication by the above matrix $\mathcal{A} = \left(\begin{smallmatrix} 3 & 2 \\ 1 & 11 \end{smallmatrix}\right)$. Since encrypting is $E(\mathbf{v}) = \mathcal{A}\mathbf{v}$, decrypting is achieved by multiplication by the inverse of $\mathcal{A}$: $D(\mathbf{v}) = \mathcal{A}^{-1}\mathbf{v}$. When is a 2×2 matrix invertible? In a first linear algebra course you learn that a (square) matrix with real entries is invertible precisely when its determinant is nonzero, and in the 2×2 case $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ has inverse

$$\frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

For matrices with entries that are integers mod m, the rule for invertibility of a matrix is *not* that the determinant is nonzero. For example, taking $m = 26$, the matrix $\left(\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right)$ has determinant $-2 \equiv 24 \bmod 26$, which is not 0 mod 26, but this matrix is not invertible mod 26: for no 2×2 matrix M is $\left(\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right)M \equiv \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \bmod 26$.

Theorem 2.1. *A 2×2 matrix with entries that are integers mod m has an inverse precisely when its determinant is invertible mod m, in which case its inverse is given by the same formula as in the real case.*

Proof. Let A be the matrix. If A has an inverse matrix B, so $AB \equiv \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \bmod m$, then taking determinants of both sides implies $(\det A)(\det B) \equiv 1 \bmod m$. Therefore $\det A$ is invertible mod m.

Conversely, if $\det A$ is invertible in mod m then the familiar inverse matrix formula from (real) linear algebra actually makes sense if we apply it to the mod m matrix A and a direct calculation shows the formula works as an inverse. □

The matrix $\mathcal{A} = \left(\begin{smallmatrix} 3 & 2 \\ 1 & 11 \end{smallmatrix}\right)$ with entries mod 26 has determinant $31 \equiv 5 \bmod 26$. The inverse of this determinant, mod 26, is 21. Therefore

$$\mathcal{A}^{-1} = 21 \begin{pmatrix} 11 & -2 \\ -1 & 3 \end{pmatrix} \equiv \begin{pmatrix} 23 & 10 \\ 5 & 11 \end{pmatrix}.$$

Call this matrix $\mathcal{B}$. (Check $\mathcal{AB} = I_2$ and $\mathcal{BA} = I_2$.) The encoded message UPFOOU has blocks

$$\begin{pmatrix} \mathrm{U} \\ \mathrm{P} \end{pmatrix} = \begin{pmatrix} 21 \\ 16 \end{pmatrix}, \quad \begin{pmatrix} \mathrm{F} \\ \mathrm{O} \end{pmatrix} = \begin{pmatrix} 6 \\ 15 \end{pmatrix}, \quad \begin{pmatrix} \mathrm{O} \\ \mathrm{U} \end{pmatrix} = \begin{pmatrix} 15 \\ 21 \end{pmatrix},$$

so we decrypt by multiplying each vector by $\mathcal{B}$ on the left:

$$\mathcal{B}\begin{pmatrix} 21 \\ 16 \end{pmatrix} = \begin{pmatrix} 19 \\ 21 \end{pmatrix}, \quad \mathcal{B}\begin{pmatrix} 6 \\ 15 \end{pmatrix} = \begin{pmatrix} 2 \\ 13 \end{pmatrix}, \quad \mathcal{B}\begin{pmatrix} 15 \\ 21 \end{pmatrix} = \begin{pmatrix} 9 \\ 20 \end{pmatrix}.$$

Putting the output blocks back together and converting them into letters gives the decrypted message:

SUBMIT.

With an encryption matrix of size 3×3 we can encrypt messages in blocks of three letters at a time. By using matrices of size 6×6 or more, attacks using frequency analysis are difficult. This technique of encryption, using matrices and modular arithmetic, was suggested by Lester Hill in the late 1920s and is called a *Hill cipher*. While the Hill cipher avoids some bad features of the Caesar cipher (*e.g.*, a letter being encrypted the same way each time), it does share one of the problems of the Caesar cipher: knowledge of the encryption and decryption functions are essentially equivalent. Anyone who knows the encryption matrix for a Hill cipher can compute its inverse (the decryption matrix) using row reduction or other methods of linear algebra. For the German Enigma machine (see Figure 1, from a display at the Computer History Museum), whose cracking by the Allies was one of the great achievements of World War II, encryption and decryption were literally the same steps. See the videos [4], [5], and [7] for illustrations of Enigma at work.

This defect of the Caesar cipher, Hill cipher, and Enigma was a feature of cryptography for most of human history: encryption and decryption functions were always effectively symmetric processes, in the sense that learning how your adversary is encrypting messages could be easily reverse-engineered to decrypt the messages. How could it be otherwise?

In the 1970s it was discovered that it *could* be otherwise: there are cryptographic algorithms based on number theory where the encryption process can be announced to the whole world and decryption in practice is still secure! RSA was one of the first practical implementations of this idea, and is based on encryption that is a power function in modular arithmetic: $E(x) = x^e \bmod m$ for suitable m and e. Decryption is also a power function: $D(y) = y^d \bmod m$ for the same modulus m and another exponent d. When we learn how RSA works, we will see that figuring out the decryption exponent d essentially requires factoring m, so the *ease* of multiplication (to take powers for encryption) and the apparent *difficulty* of factoring (to figure out the decryption exponent) keeps the system secure.

Figure 1. A German Enigma machine

3. Prelude to RSA: solving $a^n \equiv b \bmod m$ for a

To prepare ourselves for RSA, we want to understand how to undo a power computation in modular arithmetic: knowing a modulus m, positive integer n, and a power $a^n \bmod m$, can we figure out $a \bmod m$?

Example 3.1. Let's find an a satisfying $a^3 \equiv 14 \bmod 55$. Of course we could take $a = 0, 1, 2, \ldots, 54$ until we find an answer, but we want to approach the task more systematically, regarding the modulus 55 merely as a prototype for what in practice would be a very large modulus (too large to illustrate the ideas simply).

The right side of the congruence, 14, is invertible mod 55, so a has to be invertible mod 55 if it exists at all. Since $\varphi(55) = \varphi(5 \cdot 11) = (5-1)(11-1) = 40$, $a^{40} \equiv 1 \bmod 55$ by Euler's theorem, so in the power $a^n \bmod 55$, n only matters modulo $\varphi(55)$: for each $t \in \mathbf{Z}^+$,

$$a^{n+40t} = a^n(a^{40})^t \equiv a^n \bmod 55.$$

Returning to $a^3 \equiv 14 \bmod 55$, which we want to solve, raise both sides to the kth power:

$$a^{3k} \equiv 14^k \bmod 55.$$

The left side only depends on $3k \bmod 40$, so if we can choose $k \geq 1$ so that $3k \equiv 1 \bmod 40$, then $a^{3k} \equiv a^1 \equiv a \bmod 55$. To solve $3k \equiv 1 \bmod 40$ means inverting 3 modulo 40. Since

$(3, 40) = 1$, by Euclid's algorithm or other methods we can find the inverse: it is 27 mod 40. Use $k = 27$:

$$a^3 \equiv 14 \bmod 55 \Longrightarrow a^{3 \cdot 27} \equiv 14^{27} \bmod 55 \Longrightarrow a \equiv 14^{27} \bmod 55.$$

Using a computer, $14^{27} \equiv 9 \bmod 55$, so $a \equiv 9 \bmod 55$. As a check that this works,

$$9^3 = 729 = 14 + 715 = 14 + 55 \cdot 13 \equiv 14 \bmod 55.$$

To solve $a^3 \equiv 14 \bmod 55$, it was crucial that the exponent 3 is relatively prime to 40, which is $\varphi(55)$. Changing the exponent from 3 to 6, which is not relatively prime to 40, the congruence $a^6 \equiv 14 \bmod 55$ has four solutions $\pm 3, \pm 8 \bmod 55$ rather than one solution, and the congruence $a^6 \equiv 19 \bmod 55$ has no solutions.

When $(a, m) = 1$, the expression $a^n \bmod m$ depends on the base a and the exponent n in *different* ways: it depends on the base a modulo m and it depends on the exponent n modulo $\varphi(m)$. *Understand this!* The separate modular variation in the base (mod m) and the exponent (mod $\varphi(m)$), along with the apparent difficulty of figuring out $\varphi(m)$ from m if we don't know how to factor m, is the rough idea behind what makes RSA secure.

4. RSA explained

The first ingredient for RSA is a modulus that is a product of two different primes p and q. In practice they are hundreds of digits each, but for illustrative purposes we'll use much smaller p and q. Set $m = pq$. Then $\varphi(m) = (p - 1)(q - 1)$, which is easy to compute knowing p and q.

The second ingredient for RSA is an integer $e \geq 1$ such that $(e, \varphi(m)) = 1$. This will be the encryption exponent. To verify that $(e, \varphi(m)) = 1$ you could use Euclid's algorithm.

The third ingredient for RSA is an integer $d \geq 1$ such that $ed \equiv 1 \bmod \varphi(m)$, which will be the decryption exponent. It can be found by Bezout's identity if you know e and $\varphi(m)$.

Now we are ready for RSA. Having chosen your primes p and q, setting $m = pq$, and choosing an $e \geq 1$ such that $(e, \varphi(m)) = 1$, compute $d \geq 1$ such that $ed \equiv 1 \bmod \varphi(m)$. Announce to the world m and e. The numbers m and e are called your *public key.* The primes p and q and the number d are your *private key.* Messages to be sent to you should be integers x with $0 \leq x \leq m - 1$. (A longer message can be broken up into blocks that are each less than m; we'll see examples of this below.) Someone who wants to send you the message x should instead send you the encrypted message $E(x) = x^e \bmod m$. To decipher an encrypted message $y \bmod m$ that you receive, compute $D(y) \equiv y^d \bmod m$.

Example 4.1. If $p = 43$ and $q = 97$ then $m = pq = 4171$ and $\varphi(m) = 42 \cdot 96 = 4032$. Since $(5, \varphi(m)) = 1$ use $e = 5$. A solution to $5d \equiv 1 \bmod \varphi(m)$ is $d = 1613$. So $E(x) = x^5 \bmod m$ and $D(y) = y^{1613} \bmod m$ are a pair of encryption and decryption functions. For instance, if $x = 24$ then $E(x) = 24^5 \equiv 185 \bmod m$ and $D(185) = 185^{1613} \equiv 24 \bmod m$.

To show the encryption and decryption processes always undo each other, let's compute what happens if we encrypt and then decrypt a number x:

$$D(E(x)) \equiv E(x)^d \bmod m \equiv (x^e)^d \bmod m \equiv x^{ed} \bmod m,$$

so we want to verify that

$$ed \equiv 1 \bmod \varphi(m) \Longrightarrow x^{ed} \equiv x \bmod m$$

no matter what x is. Knowing $x \bmod m$ tells us x if $0 \leq x \leq m - 1$.

Theorem 4.2. *Let p and q be different prime numbers and set $m = pq$. If positive integers e and d are chosen so that $ed \equiv 1 \bmod \varphi(m)$, then for all $x \in \mathbf{Z}$,*

$$x^{ed} \equiv x \bmod m.$$

Proof. Write $ed = 1 + \varphi(m)t$ where $t \geq 0$. For every integer x,

$$x^{ed} = x^{1+\varphi(m)t} = x(x^{\varphi(m)})^t.$$

If $(x, m) = 1$ then $x^{\varphi(m)} \equiv 1 \bmod m$ by Euler's theorem, so $x^{ed} = x(x^{\varphi(m)})^t \equiv x \bmod m$.

If $(x, m) > 1$, then $x^{\varphi(m)} \not\equiv 1 \bmod m$ (a power of x can't be congruent to $1 \bmod m$ if x has a factor greater than 1 in common with m), but nevertheless we will see that it is still true that $x^{ed} \equiv x \bmod m$, or equivalently $x \cdot x^{\varphi(m)t} \equiv x \bmod m$, by using the fact that m is a product of two different primes p and q (we didn't use that m has this special form yet). In fact, the argument we give will work for all $x \in \mathbf{Z}$ in the same way, making the previous separate consideration of the case $(x, m) = 1$ superfluous.

Since p and q are relatively prime, for all $x \in \mathbf{Z}$ we have

$$x \cdot x^{\varphi(m)t} \equiv x \bmod pq \iff x \cdot x^{\varphi(m)t} \equiv x \bmod p \text{ and } x \cdot x^{\varphi(m)t} \equiv x \bmod q.$$

We have $\varphi(m) = \varphi(p)\varphi(q) = (p-1)(q-1)$, so $x \cdot x^{\varphi(m)t} = x \cdot x^{(p-1)(q-1)t}$. Therefore to check that

$$x \cdot x^{(p-1)(q-1)t} \overset{?}{\equiv} x \bmod p$$

for all $x \in \mathbf{Z}$, we can take two cases:

1) If $x \equiv 0 \bmod p$ then both sides are $0 \bmod p$, hence they are congruent.
2) If $x \not\equiv 0 \bmod p$ then $x^{p-1} \equiv 1 \bmod p$ by Fermat's little theorem, so $x^{(p-1)(q-1)t} \equiv 1 \bmod p$. Thus $x \cdot x^{(p-1)(q-1)t} \equiv x \cdot 1 \equiv x \bmod p$.

Showing $x \cdot x^{(p-1)(q-1)t} \equiv x \bmod q$ for all $x \in \mathbf{Z}$ uses the same argument, with q in place of p (take cases if $x \equiv 0 \bmod q$ and $x \not\equiv 0 \bmod q$). We are done. □

In our proof of Theorem 4.2 we started with Euler's theorem if $(x, m) = 1$, but then to handle the general case our argument only relied on Fermat's little theorem, not on Euler's theorem. In practice a random $x \bmod m$ is likely to be relatively prime to m when m is a product of two big primes, but it's good to know Theorem 4.2 works with no exceptions in x at all, even when $(x, m) > 1$, and to handle all x Euler's theorem is not used. It is a common misconception that Euler's theorem explains why encryption and decryption undo each other in RSA. *The basis for RSA is Fermat's little theorem, not Euler's theorem.* In [8, §VI], where Rivest, Shamir, and Adleman give "the underlying mathematics" behind RSA, they start by reminding the reader of Euler's theorem $x^{\varphi(m)} \equiv 1 \bmod m$ if $(x, m) = 1$, but *they never use it.* What they actually use is Fermat's little theorem in an argument having two cases, exactly as presented above.

Remark 4.3. Theorem 4.2 applies not only if m is a product of two different primes, but also if m is a product of an arbitrary number of different primes: $x^{ed} \equiv x \bmod m$ for all x when $ed \equiv 1 \bmod \varphi(m)$. If someone ever discovers a way to rapidly factor numbers that are products of two different primes then we might change RSA to use a modulus that is a product of three or more different primes (assuming an efficient technique for factoring numbers of the form pq doesn't also work on products of more than two different primes).

Example 4.4. My public key is $m = 2823907$ and $e = 3$. To send me the message $x = 71520$, encrypt it as $E(x) = x^3 \equiv 83246 \bmod m$ and send me 83246 instead.

Suppose a spy intercepts 83246. How can the spy decrypt it? The spy needs to solve $x^3 \equiv 83246 \bmod m$. To do this without a brute force search (real m have thousands of digits, not 7 digits as in this example[1]), the spy wants to invert the encryption exponent 3 mod $\varphi(m)$: if $3d \equiv 1 \bmod \varphi(m)$ then

$$D(y) = y^d \bmod m,$$

so $x \equiv D(E(x)) = 83246^d \bmod m$. Finding d requires the spy to know $\varphi(m)$, the number modulo which 3 has to be inverted; you can't solve $3d \equiv 1 \mod ??$ when the number ?? is a mystery. All the spy knows is $m = 2823907$ (and e). And that is the whole point: while it's easy for me to figure out d because I picked the primes p and q (which the spy hasn't been told) and these let me compute $\varphi(m) = (p-1)(q-1)$, there is no known way in general for the spy to figure out $\varphi(m)$ knowing only m as a plain integer but not its prime factors.

Let's reveal how m factors to see what the decryption exponent is. It turns out that $m = 1223 \cdot 2309$. I had used $p = 1223$ and $q = 2309$. Then $\varphi(m) = (p-1)(q-1) = 2820376$. A solution to $3d \equiv 1 \bmod \varphi(m)$ is $d = 1880251$, so the decryption function is $D(y) \equiv y^{1880251} \bmod m$. Check that $82346^{1880251} \equiv 71520 \bmod m$, which is consistent with the start of this example when we saw that the original message was $x = 71520$.

If I receive the new message 230748, so $E(x) = 230748$ for some unknown x, I can find x by computing $D(230748) = 230748^{1880251} \equiv 270513 \bmod m$. So $x = 270513$. As long as a spy who is eavesdropping on me can't factor m, the spy can't compute $\varphi(m)$ and thereby d, and thus can't decrypt messages to me that the spy is intercepting.

Summary of RSA: Pick two different primes p and q. Set $m = pq$ and choose $e \in \mathbf{Z}^+$ such that $(e, \varphi(m)) = 1$. The pair of numbers (m, e) is your *public key* and everyone can be told these numbers in order to encrypt a message $x \bmod m$ to you as $E(x) = x^e \bmod m$. The triple of numbers (p, q, d), where $d \in \mathbf{Z}^+$ is determined by $ed \equiv 1 \bmod \varphi(m)$, is your *private key* that you can use to decrypt messages by $D(y) \equiv y^d \bmod m$. This works since $x^{ed} \equiv x \bmod m$ for all x by Theorem 4.2, so $D(E(x)) \equiv x \bmod m$ for all $x \bmod m$. The private key has to stay secret: anyone knowing p or q knows both factors since $m = pq$ is public. From p, q, and e it's easy to find d and then decrypt messages sent to you.[2]

The first step in developing a public key is choosing two large primes, so RSA depends on being able to recognize when a large number is prime. How can we check a large number p is prime? One possibility is to run Fermat's compositeness test 20 times without finding a Fermat witness (that is, be unsuccessful in finding an a from 1 to $p-1$ such that $a^{p-1} \not\equiv 1 \bmod p$). If p passes 20 such tests we may be morally convinced p is prime and use it. But maybe p is actually a Carmichael number.[3] In practice there are better probabilistic methods of primality testing that are used, such as the Miller–Rabin test, which don't have analogues of Carmichael numbers.

[1]In 1874, William Stanley Jevons wrote [6, p. 141] "what two numbers multiplied together will produce the number 8,616,460,799? I think it unlikely that any one but myself will ever know; for they are two large primes. . .". With a computer it takes almost no time today to factor that number as $89681 \cdot 96079$. Jevons went on to say "The work would probably occupy a good computer for many weeks," but he was not referring to a machine. In the 1800s, the word "computer" referred to a *person* who did calculations.

[2]Since $E(D(x)) \equiv x^{de} \equiv D(E(x)) \bmod m$ for all x, the mathematical roles of e and d are interchangeable. This is important for the cryptographic development of digital signatures: use your private decryption exponent d to *encrypt* a confirmation message and everyone else can decrypt it using the public encryption exponent e. Nobody without knowledge of d is likely to be able to encrypt a meaningful message with d.

[3]See `https://kconrad.math.uconn.edu/blurbs/ugradnumthy/fermattest.pdf`.

The second step needed for an RSA public key, after selecting the modulus $m = pq$, is choosing the encryption exponent e, from which the decryption exponent is found by solving $ed \equiv 1 \bmod \varphi(m)$. Some care is needed here: if d (which is to be kept secret as part of the private key) is less than around $\sqrt[4]{m}$ then it is susceptible to discovery using the continued fraction expansion of e/m. This is Wiener's continued fraction attack [9].

5. Examples of RSA encryption

Let's see examples of RSA starting with a message in English text, so the process feels more like real encryption. (It's hard to be excited about encrypting a number like 4932, although that is done when you send a credit card number over the Internet.) Encrypt letters as 2-digit strings: A = 01, B = 02, and so on up to Z = 26, and let space = 27.

Example 5.1. We have MOM = 131513. Using $m = 2823907$ and $e = 3$ from Example 4.4, $E(\text{MOM}) = 131513^3 \equiv 1842379 \bmod m$. (An historically important example of a message encrypted as a sequence of numbers, not letters, is the Zimmermann telegram.)

If a message has length exceeding the modulus, *break it into smaller parts.* For modulus $m = 2823907$, which has 7 digits, a message block should have at most 3 letters (leading to a number of at most 6 digits). For modulus $m = 12964553$, which has 8 digits but leading pair of digits 12, which is less than 27, we should use a message block with at most 3 letters since for instance the letter N is encrypted as 14 and $14 > 12$; a 4-letter block that starts with N will exceed the modulus.

Example 5.2. If $m = 2823907$ and $e = 3$, then $E(x) = x^3 \bmod m$ and we saw in Example 4.4 that $D(y) = y^{1880251} \bmod m$.

The message STOP NOW is 1920151627141523 (the 27 is a space), and using blocks of length at most 3 from left to right this is $(192015, 162714, 1523)$. Apply $E(x) = x^3 \bmod m$ to each block: the first block is encrypted as $192015^3 \equiv 859833 \bmod m$, and similarly with the other two blocks. The encrypted form of this message is $(859833, 1395490, 2758917)$.

Going the other way, if we receive the encrypted message $(1294545, 1214153)$ then the original message is found by applying the decryption function to each block:

$$D(1294545) = 1294545^{1880251} \bmod m \equiv 11209 \bmod m$$

and

$$D(1214153) = 1214153^{1880251} \bmod m \equiv 2205 \bmod m.$$

As blocks of letters, the decrypted strings should have an even number of digits, but the first one has 5 digits, which means we are missing a leading zero: the original message is (011209,2205) = (ALI,VE) = ALIVE.

Remark 5.3. Pay attention to what you're doing when a message is broken into smaller parts. For example, don't break up the message in Example 5.2 into 4-letter blocks since the modulus only has 7 digits. Using the public key in Example 5.2 to encrypt STOP = 19201516 as a single block of digits, you'd get $19201516^5 \equiv 2086151 \bmod m$, but the decrypted form of $y = 2086151$ is $D(y) = y^{1880251} \equiv 2722781 \bmod m$ and $2722781 = 02722781$ can't be turned into a word since "72" and "81" do not belong to $\{01, 02, \ldots, 27\}$. We can only reasonably encrypt letters in blocks of at most 3 at a time for the public key modulus in Example 5.2.

Here are some considerations when thinking about why RSA works.

1. If $(e, \varphi(m)) > 1$ then the system need not work. For example, if $m = 217 = 7 \cdot 31$ and $e = 3$, we have $\varphi(m) = 6 \cdot 30$ and $(e, \varphi(m)) > 1$. Using $E(x) = x^3 \bmod m$, we have $8^3 \equiv 78 \bmod 217$ and $9^3 \equiv 78 \bmod 217$, so $E(8) \equiv E(9) \bmod m$. That is bad, as it means different messages could be encrypted in the same way.
2. We said in Remark 4.3 that RSA would work when the modulus m is a product of two or more different primes, not just two different primes. The primes must be different. If m has a repeated prime factor then the system need not work even if $(e, \varphi(m)) = 1$. For example, let $m = 275 = 5^2 \cdot 11$ and $e = 3$. Then $\varphi(m) = 200$, so $(e, \varphi(m)) = 1$ and a solution to $3d \equiv 1 \bmod \varphi(m)$ is $d = 67$. Using $E(x) = x^3 \bmod m$ and $D(y) = y^{67} \bmod m$, for $x = 15$ we have $D(E(x)) = (x^3)^{67} = x^{201} = 15^{201} \equiv 125 \bmod m \not\equiv 15 \bmod m$, so applying the encryption and decryption operations in succession did not return the original message.

If we know the prime factorization $m = pq$ then we can compute $\varphi(m)$ as $(p-1)(q-1)$ and solve for d in the congruence $ed \equiv 1 \bmod \varphi(m)$, thereby getting the decryption function $D(y) = y^d \bmod m$. We said at the end of Sections 2 and 3 that the security of RSA is due to the apparent difficulty of computing $\varphi(m)$ from knowing m without also knowing the prime factorization of m. *How accurate is that claim?* An astute reader might wonder if, from an RSA public key (m, e), there could be a way of computing $\varphi(m)$ without using the prime factors of m. If so, then we could solve for d in the congruence $ed \equiv 1 \bmod \varphi(m)$ and thereby get the decryption function $D(y) = y^d \bmod m$ without having to factor m. The next theorem shows that a method that lets us determine $\varphi(m)$ for an RSA modulus m leads to a method of computing the prime factors of m, so knowledge of $\varphi(m)$ and the factorization of m are basically equivalent.

Theorem 5.4. *Let $m = pq$ where p and q are different primes. If we know m and $\varphi(m)$ then we can compute p and q.*

Proof. Expand out $\varphi(m) = (p-1)(q-1) = pq - (p+q) + 1 = m - (p+q) + 1$, so $p + q = m + 1 - \varphi(m)$. Therefore if we know m and $\varphi(m)$ we know the product pq and the sum $p + q$, which up to sign are the coefficients of the quadratic polynomial with roots p and q:

$$(X - p)(X - q) = X^2 - (p+q)X + pq = X^2 - (m + 1 - \varphi(m))X + m.$$

When m and $\varphi(m)$ are known, the coefficients of the polynomial on the right side are known, so the roots of the polynomial can be found with the quadratic formula. Thus we can figure out p and q from m and $\varphi(m)$, which shows we can factor m from knowing m and $\varphi(m)$ when m is a product of two different primes. □

Example 5.5. Let $m = 3297523$ and suppose we somehow know that $\varphi(m) = 3292840$. If $m = pq$ for two different primes p and q, then by the proof of Theorem 5.4 the numbers p and q are roots of

$$X^2 - (m + 1 - \varphi(m))X + m = X^2 - 4684X + 3297523.$$

Using the quadratic formula on a computer, the roots of this polynomial are approximately 863.00000 and 3821.00000, and since the roots are supposed to be integers we expect the prime factors of m are 863 and 3821. These factors can be checked by directly multiplying them together and getting m.

To break RSA, ultimately it isn't really $\varphi(m)$ we care about, but rather the decryption exponent d. If we know an RSA public key (m, e), could there be a way of computing d

without using the factors of m? If so, then we would know $ed - 1$, which is a *multiple* of $\varphi(m)$ since $ed \equiv 1 \bmod \varphi(m)$, and $ed - 1$ is a positive integer (nobody would use $e = d = 1$ in RSA). The next theorem explains how knowing m and some positive multiple of $\varphi(m)$ lets us find the prime factors of m.

Theorem 5.6. *Let m be a product of two different odd primes and let the positive integer N be a multiple of $\varphi(m)$. Write $N = 2^r k$ where $r \geq 1$ and k is odd.*[4] *Over half of all $a \in \{2, \ldots, m-2\}$ satisfy one of the following conditions:*

(1) $1 < (a, m) < m$,

(2) $(a, m) = 1$, *the least $i \in \{0, 1, \ldots, r\}$ such that $a^{2^i k} \equiv 1 \bmod m$ is positive, and* $1 < (a^{2^{i-1}k} - 1, m) < m$.

A proof of Theorem 5.6 is omitted[5], but the next example shows how it works.

Example 5.7. We are told $m = 9{,}330{,}443$ is an RSA modulus and, somehow, we know $N = 11{,}347{,}658{,}496$ is a multiple of $\varphi(m)$. Factoring out the largest power of 2 from N, we get $N = 2^8 \cdot 44{,}326{,}791 = 2^r k$. Asking a computer for a random positive integer up to $m-1$, we are offered $a = 5{,}662{,}037$. By Euclid's algorithm, $(a, m) = 1$. Computing $a^{2^i k} \bmod m$ for $i = 0, \ldots, 8$, we find $a^{2^5 k} \equiv 4{,}259{,}023 \bmod m$ and $a^{2^6 k} \equiv 1 \bmod m$, so $i = 6$. By Euclid's algorithm, $(a^{2^5 k} - 1, m) = (4{,}259{,}022, m) = 2699$, so we have discovered a nontrivial factor of m. The complementary factor is $m/2699 = 3457$.

Theorem 5.6 provides a *probabilistic* algorithm to factor m from knowing m and a multiple of $\varphi(m)$ because a random choice of $a \in \{2, \ldots, m-2\}$ has more than a 50% chance of fitting one of the two conditions in the theorem, and both conditions lead to a nontrivial factor of m as either (a, m) or $(a^{2^{i-1}k} - 1, m)$. When $m = pq$, a nontrivial factor of m is either p or q, so knowing one of them tells us both of them. It should not take more than a few random values of a before we get one that leads to a nontrivial factor of m by the algorithm in Theorem 5.6.

The only type of RSA modulus that Theorem 5.6 does not apply to is an even one: $m = 2q$ for an odd prime q. This is irrelevant in practice since nobody would use an RSA modulus with 2 as a prime factor.

If we know an RSA public key (m, e) then Theorem 5.6 tells us that determining either d or the prime factorization of m are practically equivalent computational tasks: we get d from p and q by using Euclid's algorithm to solve for d in $ed \equiv 1 \bmod (p-1)(q-1)$, and we get p and q from d by using the probabilistic algorithm in Theorem 5.6 with $N = ed - 1$ to find a nontrivial factor of m.

Cryptographic protocols like the Caesar cipher, the Hill cipher, and the German Enigma machine, are called *symmetric* since the encryption and decryption processes can each be determined from the other one in a computationally efficient way (in the case of Enigma messages, encryption and decryption were exactly the same). Using a symmetric cipher requires keeping the encryption operation hidden from adversaries. By comparison, RSA is called an *asymmetric* cipher since knowing how to encrypt does not – as far as we can tell – easily tell us how to decrypt. Asymmetric ciphers were created for the first time in

[4]We have $r \geq 1$ since $\varphi(m)$ is even, so its multiple N is also even.

[5]Theorem 5.6 is in fact true for all odd $m > 1$ that are not prime powers, not just RSA moduli pq. A proof of the theorem is based on ideas similar to what is needed to justify the Miller–Rabin primality test. See Corollary A.2 in `https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf` with $\varphi(n) = 2^e k$ there replaced by $N = 2^r k$ here.

the 1970s, with RSA being one of the earliest examples. The term for ciphers where the data needed to do encryption can be made public without sacrificing the security of the decryption process is "public key cryptography."

An asymmetric cipher may appear to be a superior method of encryption compared to a symmetric cipher, but the reality is that both techniques are used together. Modern symmetric ciphers run much faster than asymmetric ciphers (especially on the decryption end), so RSA is not convenient to use for the exchange of long messages. In practice RSA is used for two parties to exchange a small piece of information (called a "key exchange") that can be used to set up communication through a symmetric cipher like the Advanced Encryption Standard. Since the information needed to establish a common symmetric cipher was transmitted through RSA, the encryption function for the symmetric cipher has not been revealed to the public and is safe to use for that one-time session.

6. History of RSA

The RSA algorithm was created in 1977 and first publicized in Martin Gardner's *Scientific American* column that year [3]. Nobody knew at the time that the idea had essentially already been developed by British intelligence a few years before. Clifford Cocks, who worked for GCHQ (Government Communications Headquarters, which is the British counterpart to the NSA), had studied number theory as a student at Cambridge and proposed in a classified report in 1973 exactly the RSA algorithm with the choice $e = m$. That is, Cocks took the encryption exponent to be the modulus, so he had to assume the primes p and q satisfy $(pq,(p-1)(q-1)) = 1$, or equivalently $(p, q-1) = 1$ and $(q, p-1) = 1$ since obviously p is relatively prime to $p-1$ and q is relatively prime to $q-1$. This is less flexible than RSA, but the underlying math is the same. Due to the lack of powerful enough computers at the time, GCHQ did not develop this work into a practical system before it was rediscovered by Rivest, Shamir, and Adleman. The original work at GCHQ was declassified in 1997, 20 years after the public (re)discovery of RSA.

At GCHQ the name that had been used in place of "public key cryptography" was "nonsecret encryption," which is how Cocks refers to it in the title of his report [1]. An historical summary of how this idea was developed at GCHQ has been written by James Ellis [2].

References

[1] C. Cocks, A Note on Non-Secret Encryption, CESG report, November 20, 1973. URL http://cryptocellar.org/cesg/notense.pdf.

[2] J. H. Ellis, The Story of Non-Secret Encryption, CESG Report, 1987. URL https://cryptocellar.org/cesg/possnse.pdf.

[3] M. Gardner, A new kind of cipher that would take millions of years to break, *Scientific American* **237** (Aug. 1977), 120–124.

[4] J. Grime, Enigma Machine - Numberphile, https://www.youtube.com/watch?v=G2_Q9FoD-oQ.

[5] J. Grime, Flaw in the Enigma Code - Numberphile, https://www.youtube.com/watch?v=V4V2bpZlqx8.

[6] W. S. Jevons, "The Principles of Science: a Treatise on Logic and Scientific Method," MacMillan & Co., New York, 1874. URL https://archive.org/stream/principlesofscie00jevorich#page/n165/mode/2up.

[7] J. Owen, How did the Enigma Machine work? https://www.youtube.com/watch?v=ybkkiGtJmkM.

[8] R. L. Rivest, A. Shamir, L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Comm. ACM* **21**, Feb. 1978. URL https://people.csail.mit.edu/rivest/pubs/RSA78.pdf.

[9] M. Wiener, Cryptanalysis of short RSA secret exponents, *IEEE Transactions on Info. Theory* **36** (1990), 553-558.