Contents

1 Classical Cryptosystems and Modular Arithmetic

In this chapter, we discuss the basic ideas of cryptography and a number of historical cryptosystems. In order to treat these systems effectively we need to develop some basic results from number theory, primarily the basics of modular arithmetic, so we will take a lengthy digression to do that before returning to cryptography.

Most modern cryptography makes heavy use of modular arithmetic and number theory, and most of these methods rely on the assumed difficulty of solving one or more problems in number theory, such as computing discrete logarithms, factoring large integers, and computing square roots modulo composite integers.

We will first discuss a few pre-modern cryptosystems, pointing out the flaws that ultimately led to their abandonment.

1.1 General Principles of Cryptography

- Cryptography is the name given to encoding and transmitting information in a way that makes it difficult for someone else to intercept and use.
    - Many of the earliest uses of cryptography were to send secure military information and orders that could not be decoded by enemy forces.
    - In the modern setting, secure cryptography is at the heart of internet commerce: for example, it allows merchants and credit card companies to exchange purchasing information without anyone else being able to eavesdrop.
- In analysis of cryptography, it is useful to have a standard list of placeholder names:
    - Alice and Bob refer to two parties attempting to exchange information. (Generally, Alice wants to send a message to Bob, though the communication can be two-directional.)

- Eve refers to a non-malicious eavesdropper, who can listen in to the communications between Alice and Bob, but will not alter them.
- Mallory refers to a malicious eavesdropper, who can listen to Alice and Bob's communications and may also attempt to impersonate them or alter their messages.

- In general, a cryptographic system works as follows:
  - Alice wishes to send a secure message to Bob.
  - Alice takes her unencrypted message, her plaintext, and encrypts it somehow to obtain a ciphertext.
  - Alice then sends the ciphertext to Bob, who then decodes it to recover Alice's original message.

- We will generally write plaintexts in **bold lowercase** and ciphertexts in **BOLD UPPERCASE**.
  - Note: For the ease of readability, when it is reasonable we will include spaces (becauseitishardtoreadalengthytextwithnospaces) when rendering plaintexts and ciphertexts, but when we encode messages we will not use the spaces.

- An effective cryptosystem must strike a reasonable balance between the following:
  1. Being straightforward enough that Alice can easily encode the message.
  2. Transmitting the information faithfully so that the ciphertext can be decoded correctly by Bob.
  3. Being secure enough that, if Eve intercepts the ciphertext, she cannot recover the original plaintext.

- There are other kinds of attacks on a simple messaging system that cryptography can also addess:
  1. Eve could want to decipher just the one message Alice sent, or she may want a method for reading all messages that Alice has sent. If Alice's procedure is not secure enough, it is possible that if Eve can decode a single message then she can decode all of Alice's other messages too.
  2. A more malicious individual, Mallory, may want not only to read the message, but to alter it before passing it on to Bob in such a way that Bob thinks Alice sent the altered message.
  3. Mallory may also want to pretend to be Bob and communicate with Alice in a way that makes it seem as if she is actually communicating with Bob.

- Historically, most cryptography relied on making the message appear nonsensical and unreadable, or by hiding it in some other more innocuous location (e.g., by encoding the message in the first letter of each word in a document).
  - This latter procedure is sometimes called steganography, the hiding of secret information in plain sight. It is also interesting, but is not really the purpose of cryptography.
  - The security of steganography comes only from the obscurity of the method used: once an adversary learns of the procedure being used to hide the information, it is trivial for them to obtain it.
  - The true strength of cryptography comes from the fact that it is possible to design procedures that, even if Eve has full knowledge of how Alice encoded her message, Eve still cannot decode it.

- One of the most classical cryptosystems is the Caesar shift algorithm (so named because it was used by Julius Caesar): simply shift each letter of the plaintext forward a fixed number of letters in the alphabet (wrapping around from Z to A, as needed).
  - Thus, shifting the plaintext **evil hal** 1 letter forward produces **FUJM IBM**.
  - The Caesar shift is quite easy to break for a number of reasons, the most obvious of which is that each message has only 26 different possible encodings, and all of them can be rapidly written down by shifting the encrypted message forward by 1 letter 25 separate times.

- We will also mention a few different types of attacks on cryptosystems:
  - Ciphertext-only attack: Eve only has a copy of the ciphertext and wants to decode it.

- Known-plaintext attack: Eve has a copy of the ciphertext and the associated plaintext. In this case Eve's goal is to break the encryption system so she can read future ciphertexts that are encoded using the same system.
- Chosen-plaintext attack: Eve is able to choose a plaintext and see how it encodes to a ciphertext. (For example, if the encryption algorithm is implemented as software on a computer, Eve would have access to the part of the program that encodes messages.) Again, Eve's goal is to try to break the encryption algorithm so she can read future ciphertexts.
- Chosen-ciphertext attack: Eve is able to choose a ciphertext and see how it decodes to a plaintext. (For example, if the encryption algorithm is implemented as software on a computer, Eve would have access to the part of the program that decodes messages.)

1.2 The Integers and Modular Arithmetic

- In order to analyze classical cryptosystems, we need to develop some results from modular arithmetic. To do this in a reasonable manner we first require a few basic properties of the integers.

1.2.1 The Integers, Divisors, Common Divisors

- Recall that the set of integers $\mathbb{Z}$ consists of the natural numbers $\mathbb{N}$ (1, 2, 3, 4, ...), along with their negatives (-1, -2, -3, -4, ...) and zero (0).
  - The set of integers comes attached with a number of natural binary operations: addition $+$, subtraction $-$, and multiplication $\cdot$, which should surely be very familiar.
  - It is not always possible to divide one integer by another and obtain an integer result: there is no integer n such that $1/2 = n$, for example.
  - Rather than trying to define the division operation (which ultimately leads to the construction of the set of rational numbers $\mathbb{Q}$) we will instead focus on the idea of "divisibility".
- Definition: If $a \neq 0$, we say that a divides b, written $a|b$, if there is an integer k with $b = ka$.
  - Examples: $2|4$, $(-7)|7$, and $6|0$.
- There are a number of basic properties of divisibility that follow immediately from the definition and properties of arithmetic:
  - If $a|b$, then $a|bc$ for any c.
  - If $a|b$ and $b|c$, then $a|c$.
  - If $a|b$ and $a|c$, then $a|(xb + yc)$ for any x and y.
  - If $a|b$ and $b|a$, then $a = \pm b$.
  - If $a|b$, and $a, b > 0$, then $a \leq b$.
  - For any $m \neq 0$, $a|b$ is equivalent to $(ma)|(mb)$.
- If $0 < b < a$ and b does not divide a, we can still attempt to divide a by b to obtain a quotient and remainder: this is simply the statement that the long-division algorithm familiar from elementary school actually works. Formally:
- Proposition (Quotient With Remainder): If a and b are positive integers, then there exist unique integers q and r such that $a = qb + r$ with $0 \leq r < b$. Furthermore, $r = 0$ if and only if $b|a$.
  - This result is not straightforward to justify without a careful construction of the integers. We will simply take it as a basic fact.
- Definition: If $d|a$ and $d|b$, then d is a common divisor of a and b. If a and b are not both zero, then there are only a finite number of common divisors: the largest one is called the greatest common divisor, or gcd, and denoted by $\gcd(a, b)$.

- Warning: Many authors use the notation (a, b) to denote the gcd of a and b: this stems from the notation used for ideals in abstract algebra. We will always write gcd explicitly, since otherwise it is easy to confuse the gcd with an ordered pair (a, b).
- Example: The positive divisors of 30 are 1, 2, 3, 5, 6, 10, 15, 30. The positive divisors of 42 are 1, 2, 3, 6, 7, 14, 21, 42. The common (positive) divisors are 1, 2, 3, and 6, and so $\gcd(30, 42) = 6$.

- Definition: If $\gcd(a, b) = 1$, we say a and b are relatively prime.
  - Example: 7 and 12 are relatively prime, but 36 and 92 are not relatively prime.

- We can deduce a number of basic facts about greatest common divisors:
  - If $m > 0$, then $\gcd(ma, mb) = m \cdot \gcd(a, b)$.
  - If $d > 0$ divides both a and b, then $\gcd(a/d, b/d) = \gcd(a, b)/d$.
  - If a and b are both relatively prime to m, then so is ab.
  - For any integer x, $\gcd(a, b) = \gcd(a, b + ax)$.
  - If $c|ab$ and b, c are relatively prime, then $c|a$.

- Computing the greatest common divisor by writing down all the divisors of each number is very inefficient. There is a much faster method:

- Theorem (Euclidean Algorithm): Given integers $0 < b < a$, repeatedly apply the division algorithm as follows, until a remainder of zero is obtained:

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

  Then $d = \gcd(a, b)$ is equal to the last nonzero remainder, r_{k+1}. Furthermore, by successively solving for the remainders and plugging in the previous equations, r_{k+1} can be explicitly written as an integer linear combination of a and b, meaning that there exist integers x and y such that $d = xa + yb$.
  - The Euclidean Algorithm is an extremely fast way to compute the greatest common divisor: it does not require factoring of any kind, and will compute the gcd of two $\leq n$-digit integers after at most $1 + 4 \log_2 n$ iterations.
  - The idea behind the Euclidean algorithm is that if $d|a$ and $d|b$, then $d|(a - q_1 b)$: so we see that $d|r_1$ and $d|b$. We can then repeat the process to see that $d|r_2$, $d|r_3$, and in general d will divide the remainder at each step. The sequence of remainders will necessarily decrease at each stage, and so we must eventually reach a remainder of zero. So any common divisor of a, b will divide the last nonzero remainder: this necessarily means that this last nonzero remainder is the greatest common divisor.

- Example: Find the gcd of 30 and 42 using the Euclidean Algorithm, and write the gcd explicitly as a linear combination of 30 and 42.
  - First, we divide:

$$\begin{aligned} 42 &= 1 \cdot 30 + 12 \\ 30 &= 2 \cdot 12 + \boxed{6} \\ 12 &= 2 \cdot 6. \end{aligned}$$

    Since 6 is the last nonzero remainder, it is the gcd.

- For the linear combination, we solve for the remainders:

$$\begin{aligned} 12 &= 42 - 1 \cdot 30 \\ 6 &= 30 - 2 \cdot 12 = 30 - 2 \cdot (42 - 1 \cdot 30) = 3 \cdot 30 - 2 \cdot 42 \end{aligned}$$

  so we obtain $\boxed{6 = 3 \cdot 30 - 2 \cdot 42}$.

- We also have a notion of a common multiple:
- Definition: If $a|l$ and $b|l$, l is a common multiple of a and b. Among all (nonnegative) common multiples of a and b, the smallest such l is called the least common multiple of a and b.
  - Example: The least common multiple of 30 and 42 is 210.
  - The least common multiple is often mentioned in elementary school in the context of adding fractions (for finding the "least common denominator").
- The least common multiple has fewer nice properties than the gcd. It does obey the relation $m \cdot \text{lcm}(a, b) = \text{lcm}(ma, mb)$.
- Proposition: If $a, b > 0$, the gcd and lcm satisfy $\gcd(a, b) \cdot \text{lcm}(a, b) = ab$.
  - Thus, if we want to calculate the lcm of two arbitrary integers, we can just compute the gcd using the Euclidean Algorithm, and then apply the result of this proposition to get the lcm.
  - Proof: First suppose a and b are relatively prime, and let l be a common multiple. Since $a|l$ we can write $l = ak$ for some integer k: then since $b|ak$ and $\gcd(a, b) = 1$, we conclude by properties of divisibility that $b|k$, meaning that $k \geq b$ and thus $l \geq ab$. But clearly ab is a common multiple of a and b, so it is the least common multiple.
  - In the general case, let $d = \gcd(a, b)$. Then $\gcd(a/d, b/d) = 1$, so by the above we see that $\text{lcm}(a/d, b/d) = ab/d^2$. Then $\gcd(a, b) \cdot \text{lcm}(a, b) = d \cdot d\,\text{lcm}(a/d, b/d) = ab$, as desired.

1.2.2 Primes and Prime Factorization

- Definition: If $p > 1$ is an integer, we say it is prime if there is no d with $1 < d < p$ such that $d|p$: in other words, if p has no positive divisors other than 1 and itself. If $n > 1$ is not prime, we say it is composite. (The integer 1 is neither prime nor composite.)
  - The first few primes are 2, 3, 5, 7, 11, 13, 17, 19, and so forth.
- Here is a basic fact about prime numbers that in more advanced contexts is often used as the actual definition of a prime number:
- Proposition: If p is prime and $p|ab$, then $p|a$ or $p|b$.
  - Proof: If $p|a$ we are done, so assume that $p \nmid a$.
  - Consider $\gcd(a, p)$: it divides p, hence is either 1 or p, but it is not p because p does not divide a.
  - Therefore, $\gcd(a, p) = 1$, so a and p are relatively prime. Then since $p|ab$ and a, p are relatively prime, we see that $p|b$ from our earlier results.
- The prime numbers are the basic building blocks of the integers under multiplication. The most fundamental fact about primes is that every integer can be written as a product of primes in an essentially unique way:
- Theorem (Fundamental Theorem of Arithmetic): Every positive integer can be factored into a product of primes, and this prime factorization is unique up to reordering of the factors.
  - As an example, the prime factorization of 60 is $60 = 2 \cdot 2 \cdot 3 \cdot 5$.
  - Proof: To show that every positive integer can be written as a product of primes, we use strong induction.

- The result clearly holds if $n = 1$ since 1 is an empty product. Now suppose $n > 1$. If n is prime, we are done, so assume that n is not prime, so it is composite. By definition, there exists a d with $1 < d < n$ such that $d|n$: then n/d is an integer satisfying $1 < n/d < n$. By the strong induction hypothesis, both d and n/d can be written as a product of primes; multiplying these two products then yields n as a product of primes.
- To show that the factorization is unique, suppose that n is the smallest positive integer that has two different factorizations: $n = p_1 p_2 \cdots p_k = q_1 q_2 \cdots q_l$. If any of the primes p_i and q_j were equal, we could cancel the corresponding terms and obtain a smaller n, so $p_1 \neq q_j$ for any j with $1 \leq j \leq l$.
- But since p_1 is prime and divides $q_1 q_2 \cdots q_l$, by repeated application of the previous proposition we see that p_1 must divide one of $q_1, q_2, \ldots, q_l$: say, q_i. But the only divisors of q_i are 1 and q_i, and p_1 cannot be either of them. This is a contradiction, and we are done.

- To save space, we group equal primes together when actually writing out the canonical prime factorization of a positive integer: thus, $12 = 2^2 \cdot 3$, $720 = 2^2 \cdot 3^2 \cdot 5$, and so forth. More generally, we often write the prime factorization in the form $n = \prod p_i^{n_i}$, where the p_i are some (finite) set of primes and the n_i are their corresponding exponents.
- We can also describe divisibility and the greatest common divisor using prime factorizations.
- Proposition: If $a = \prod p_i^{a_i}$ and $b = \prod p_i^{b_i}$, then $a|b$ if and only if $a_i \leq b_i$ for each i. In particular, $\gcd(a, b) = \prod p_i^{\min(a_i, b_i)}$, and $\text{lcm}(a, b) = \prod p_i^{\max(a_i, b_i)}$.
  - Proof: We observe that if $b = ak$ and $k = \prod p_i^{k_i}$, then $a_i + k_i = b_i$. Since all exponents are nonnegative, saying that such an integer k exists is equivalent to saying that $a_i \leq b_i$ for all i.
  - The statements about the gcd and lcm then follow immediately, since (for example) the exponent of p_i in the gcd is the largest integer that is $\leq a_i$ and $\leq b_i$, which is a more convoluted way of saying the minimum of a_i and b_i.

- One question we might have is: how many primes are there? The most basic answer to this question is that there are infinitely many primes:
- Theorem (Euclid): There are infinitely many prime numbers.
  - Proof: Suppose there are only finitely many prime numbers $p_1, p_2, \ldots, p_k$, and consider $n = p_1 p_2 \cdots p_k + 1$.
  - Since n is bigger than each p_i, n cannot be prime (since it would necessarily have to be on the list).
  - Therefore n is composite. Consider the prime factorization of n: necessarily at least one prime on the list must appear in it: say p_i.
  - Since p_i also divides $p_1 p_2 \cdots p_k$, we see that p_i therefore divides $n - p_1 p_2 \cdots p_k = 1$. But this is a contradiction. Hence there are infinitely many primes.

- A natural followup question to Euclid's theorem that there infinitely many prime numbers is: how common are the primes? This result lies much deeper:
- Theorem (Prime Number Theorem): Let $\pi(n)$ be the number of primes p with $1 \leq p \leq n$. Then $\pi(n)$ is approximately equal to $\frac{n}{\ln(n)}$: more specifically, $\lim_{n \to \infty} \frac{\pi(n)}{n/\ln(n)} = 1$.

1.2.3 Modular Arithmetic and Congruences

- The ideas underlying modular arithmetic are familiar to anyone who can tell time.
  - For example, 3 hours after 11 o'clock, it is 2 o'clock, despite the fact that $3 + 11$ is 14, not 2: we identify times that are 12 hours apart as the same time of day.

- Modular arithmetic is simply a formalization of this "clock arithmetic":

- Definition: If m is a positive integer and m divides $b-a$, we say that a and b are congruent modulo m (or equivalent modulo m), and write "$a \equiv b \pmod{m}$".
  - Observe that if $m|(b-a)$, then $(-m)|(b-a)$ as well, so we do not lose anything by assuming that the modulus m is positive.
  - In general, the statement $a \equiv b \pmod{m}$ can be thought of as saying "a and b are equal, up to a multiple of m".
  - Notation: As shorthand we often also write "$a \equiv b \pmod{m}$", or even just "$a \equiv b$" (when the modulus m is clear from the context).
  - Example: $3 \equiv 9 \pmod{6}$, since 6 divides $9-3=6$.
  - Example: $-2 \equiv 28 \pmod{5}$, since 5 divides $28-(-2)=30$.
  - Example: $0 \equiv -666 \pmod{3}$, since 3 divides $-666-0=-666$.
  - If m does not divide $b-a$, we say a and b are not congruent mod m, and write $a \not\equiv b \pmod{m}$.
  - Example: $2 \not\equiv 7 \pmod{3}$, because 3 does not divide $7-2=5$.
- Modular congruences behave quite similarly to equalities:
  - For any a, $a \equiv a \pmod{m}$.
  - For any a and b, $a \equiv b \pmod{m}$ if and only if $b \equiv a \pmod{m}$.
  - For any a, b, c, if $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, then $a \equiv c \pmod{m}$.
  - For any a, b, c, d, if $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a+c \equiv b+d$ and $ac \equiv bd \pmod{m}$.
- The properties above say that we can do addition and multiplication with congruences in the same way as we can with equalities, provided the modulus m is always the same.
  - Example: Observe that $7 \equiv 3 \pmod{4}$ and $1 \equiv -11 \pmod{4}$. If we add these congruences we get $8 \equiv -8 \pmod{4}$ which is true, and if we multiply them we get $7 \equiv -33 \pmod{4}$, which is also true.
- Here are a few more properties of congruences that essentially follow directly from the definition:
  - For any a, b, c with $c > 0$, if $a \equiv b \pmod{m}$, then $ac \equiv bc \pmod{mc}$.
  - For any a, b, d with $d|m$, if $a \equiv b \pmod{m}$, then $a \equiv b \pmod{d}$.
  - For any a, b, k with $k > 0$, if $a \equiv b \pmod{m}$ then $a^k \equiv b^k \pmod{m}$.
  - For any a, b, if $p(x)$ is a polynomial with integer coefficients, then $a \equiv b \pmod{m}$ implies $p(a) \equiv p(b) \pmod{m}$.
- Definition: If a is an integer, and $a \equiv b \pmod{m}$, we say that b is a residue of a mod m. The residue class of a modulo m, denoted $\overline{a}$, is the collection of all integers congruent to a modulo m. Observe that $\overline{a} = \{a + km,\ k \in \mathbb{Z}\}$.
  - The division algorithm says that for any $a \in \mathbb{Z}$ there exists a unique r with $0 \leq r < m$ such that $a = qm + r$ with $q \in \mathbb{Z}$. In other words: every integer is congruent to precisely one of the m integers 0, 1, ... , $m-1$.
- Using the residue classes, we can write down addition and multiplication tables modulo m.
  - Our properties above say that, if we want to compute $a+c$ modulo m, then no matter which element b in the residue class of a and which element d in the residue class of c we take, the sum $b+d$ will lie in the same residue class as $a+c$, and the product bd will lie in the same residue class as ac.
  - Thus, everything makes perfectly good sense if we label the residue classes with the integers 0 through $m-1$ and simply do the arithmetic with those residue classes.

- Here are the addition and multiplication tables modulo 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

$\cdot$	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

- An alternative way we could attempt to describe the arithmetic modulo m would be only to allow ourselves to work with the integers 0 through $m-1$, and simply "reduce modulo m" at the end of every computation to obtain a result lying in this range.
  - Thus, for example, to compute $3+10$ modulo 12, we would add to get 13 and then we reduce obtain 1 modulo 12. Similarly, we would compute $3 \cdot 10 = 30$, and then reduce to say $3 \cdot 10 \equiv 6 \bmod 12$.
  - However, this is a rather cumbersome and inelegant description. This definition is often used in programming languages, where "a mod m", frequently denoted "$a\%m$", is defined to be a *function* returning the corresponding remainder in the interval $[0, m-1]$.
  - Observe that with this definition, it is not true that $(a+b)\%m = (a\%m) + (b\%m)$, nor is is true that $ab\%m = (a\%m) \cdot (b\%m)$, since the sum product may each exceed m. Instead, to obtain an actually true statement, one would have to write something like $ab\%m = [(a\%m) \cdot (b\%m)]\%m$.
  - In order to avoid such horrible kinds of statements, the best viewpoint really is to think of the statement $a \equiv b \pmod m$ as a congruence that is a "weakened" kind of equality, rather than always reducing each of the terms to its residue in the set $\{0, 1, \ldots, m-1\}$.
- Although modular congruences share a number of properties with (standard) equality, there are some very important differences.
  - For example, if a, b, c are any real numbers with $a \neq 0$, then we can cancel a from the relation $ab = ac$ to conclude that $b = c$.
  - We cannot always cancel so nicely with congruences: for example, $2 \cdot 1$ is congruent to $2 \cdot 10$ modulo 6, but 1 is not congruent to 10 modulo 6.
  - The issue here is that 2 and the modulus 10 have a common divisor.
- Proposition: If $m > 0$ and $d = \gcd(a, m)$, then $ax \equiv ay \pmod m$ is equivalent to $x \equiv y \pmod{m/d}$.
  - Proof: First suppose $ax \equiv ay \pmod m$. Then there exists an integer k with $a(y-x) = km$, so $\frac{a}{d}(y-x) = \frac{m}{d}k$. Since $\gcd(a/d, m/d) = \gcd(a, m)/d = 1$, we see that $\frac{m}{d}$ divides $y - x$, meaning that $x \equiv y \pmod{m/d}$.
  - For the other direction, suppose $y - x = \frac{m}{d}k$. Then $a(y-x) = \frac{a}{d}mk$, and since $d|a$, we see that m divides $a(y-x)$, meaning that $ax \equiv ay \pmod m$.
- Corollary: If a and m are relatively prime, $ax \equiv ay \pmod m$ implies $x \equiv y \pmod m$.
  - This result is an immediate consequence of the previous one (simply take $d = 1$), but we will invoke it often enough that it is worth explicitly writing down.
  - We can view this statement as saying that we can "divide by a" on both sides of this congruence.
- It is reasonable to wonder, more generally, when exactly we are allowed to divide both sides of a congruence by something: or in other words, which elements have multiplicative inverses?
- Definition: We say a is a unit modulo m if it has a multiplicative inverse: that is, if there is some b such that $ab \equiv 1 \pmod m$.
  - The multiplicative inverse b is often written as $a^{-1} \pmod m$, or even sometimes as $1/a \pmod m$.
  - Example: From the multiplication table modulo 5, we see that 1, 2, 3, and 4 all have multiplicative inverses, which are 1, 3, 2, and 4 respectively, but 0 does not. So 1, 2, 3, and 4 are the units modulo 5.

- Example: Modulo 6, it is straightforward to check that the only units are 1 and 5 (whose multiplicative inverses are 1 and 5 respectively).
- Example: Modulo 14, the units turn out to be 1, 3 (inverse 5), 5 (inverse 3), 9 (inverse 11), 11 (inverse 9), and 13.
- Remark: Technically, the definition does not require that there be only one such b. However, there can be only one such b: if $ab_1 \equiv 1 \equiv ab_2 \pmod{m}$, then $b_1 \equiv b_1ab_2 \equiv b_2 \pmod{m}$.

- Based on the examples, it seems that the units modulo m are precisely those integers which are relatively prime to m.
- Proposition (Units Modulo m): An integer a is a unit modulo m if and only a and m are relatively prime.
  - Proof: First suppose a is a unit modulo m. Then there exists an integer b with $ab \equiv 1 \pmod{m}$. Equivalently, there exists a k such that $ab - km = 1$. But $\gcd(a, m)$ divides any linear combination of a and m, so it must equal 1.
  - Now suppose $\gcd(a, m) = 1$. Then, by the Euclidean algorithm, we can find integers x and y such that $xa + ym = 1$. Reducing modulo m yields $xa \equiv 1 \pmod{m}$, so x is the multiplicative inverse of a modulo m.
- The above proposition also gives a method to compute the inverse of a (presuming it has one): simply apply the Euclidean algorithm to generate x and y with $xa + ym = 1$: then the inverse of a modulo m is x.
- Example: Check that 7 is a unit modulo 52, and then find its multiplicative inverse.
  - We apply the Euclidean algorithm:

$$\begin{aligned} 52 &= 7 \cdot 7 + 3 \\ 7 &= 2 \cdot 3 + 1 \end{aligned}$$

    so the gcd is indeed 1.
  - Now we have $3 = 52 - 7 \cdot 7$ and then $1 = 7 - 2 \cdot 3 = 7 - 2 \cdot (52 - 7 \cdot 7) = 15 \cdot 7 - 2 \cdot 52$.
  - Then $15 \cdot 7 \equiv 1 \pmod{52}$, so $7^{-1} = \boxed{15 \pmod{52}}$.

1.3 Shift Ciphers, Affine Ciphers, and Substitution Ciphers

- We now return to our discussion of cryptography: we will revisit the Caesar shift using the perspective of modular arithmetic, and then discuss some more general variants.
- To describe the Caesar shift, we will associate letters to numbers. We will adopt the convention that a corresponds to 0, b corresponds to 1, ... , and z corresponds to 25.
- Mathematically we can describe the Caesar shift as follows (where we automatically convert the letters of the message to numbers):
  - First, we choose a key k.
  - We encrypt the message by applying the function $f(x) = x + k \pmod{26}$ to the numbers making up the message.
  - To decrypt, we apply the inverse function $f^{-1}(x) = x - k \pmod{26}$ to the encrypted message.
- Let us analyze the security (or, really, the lack of security!) of the Caesar shift cryptosystem:
  - Suppose Eve has only the ciphertext to work with. The simplest approach is for her to write down all 26 possible decodings, since there are only 26 possible keys. Assuming that the message has more than a few characters, it is very unlikely that there is more than one possible decoding that makes any sense.
  - If Eve has access to any one character of the plaintext in addition to the ciphertext, she needs to do even less work, because she immediately determine the key and decode the message since the key is simply the difference between the ciphertext and plaintext characters.

- We can generalize the shift cipher by using a more complicated encoding function. Let us instead consider the class of linear encoding functions of the form $f(x) = ax + b \pmod{26}$ for some choices of a and b. These functions are affine functions, so the associated cipher is called an affine cipher.
  - In order for this encoding to be invertible, we need there to be an inverse function $f^{-1}(x)$.
  - Over the rational numbers (or the reals) the inverse of $f(x) = ax + b$ is $f^{-1}(x) = a^{-1}(x - b)$.
  - So in order for this function to be invertible, we need a to be a unit modulo 26: otherwise, there will be several different numbers which encrypt to the same value (making it impossible to decode the message properly.)
- Here is the formal description of the affine cipher:
  - First, we choose a key, which is a pair (a, b) where a is a unit modulo 26.
  - We encrypt a message by applying the function $f(x) = ax + b \pmod{26}$ to the numbers making up the message.
  - To decrypt, we apply the inverse function $f^{-1}(x) = a^{-1}(x - b) \pmod{26}$ to the encrypted message.
- We can make a table for the encryption of each letter under the affine cipher to save time when encoding long messages.
  - For example, here is the encoding table for the affine cipher $f(x) = 7x + 3 \pmod{26}$:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
3	10	17	24	5	12	19	0	7	14	21	2	9	16	23	4	11	18	25	6	13	20	1	8	15	22
d	k	r	y	f	m	t	a	h	o	v	c	j	q	x	e	l	s	z	g	n	u	b	i	p	w

  - So, for example, the encoding of the plaintext **hydrogeology** is **APYSXTFXCXTP**.
  - To compute the inverse function, we need to evaluate 7^{-1} modulo 26. Using the Euclidean algorithm we can derive the relation $1 = 3 \cdot 26 - 11 \cdot 7$, so $7^{-1} \equiv -11 \equiv 15 \pmod{26}$.
  - Therefore, the inverse function is $f^{-1}(x) = 15(x - 3) \equiv 15x + 7 \pmod{26}$.
  - Here is the corresponding table for this function, which we can check will undo the encoding in the table above (as of course it should):

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
7	22	11	0	15	4	19	8	23	12	1	16	5	20	9	24	13	2	17	6	21	10	25	14	3	18
h	w	l	a	p	e	t	i	x	m	b	q	f	u	j	y	n	c	r	g	v	k	z	o	d	s

- The affine cipher has a bit more cryptographic depth than the shift cipher, but not a great deal more. Let us study some possible attacks:
  - Attack 1, ciphertext-only: If Eve has access only to a ciphertext, one approach would be via brute force, by writing down all of the possible decodings.
    - It is reasonably easy to write down a list of the possible values of a, of which there are 12: $\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$. Since each value of a has 26 possible values of b, there are a total of $26 \cdot 12 = 312$ possible keys. (Note that 26 of these have $a = 1$, and these are Caesar shifts.)
    - It would only be worth doing a brute-force attack on an affine cipher by hand for an extremely important message, but it is still very easy to do write down all 312 possibilities with a computer.
  - Attack 2, known-plaintext: If Eve has access to a plaintext and its associated ciphertext, she can compare corresponding letters and, with reasonable care, she can determine the key.
    - Each pair (p, c) of letters in the plaintext and ciphertext determine an equation $f(p) = c$, which is a linear relation $c = ap + b$.
    - In general, since two points are needed to determine a line, Eve will need to know two corresponding pairs of letters in the plaintext and ciphertext to be able to determine the key directly. (If she only knows one pair, she will end up with a list of 12 possible solutions to check, one for each possible a.)

    - Explicitly, if Eve knows the pairs (p_1, c_1) and (p_2, c_2), she gets the equations $c_1 \equiv ap_1 + b \pmod{26}$ and $c_2 \equiv ap_2 + b \pmod{26}$. Subtracting these gives $c_2 - c_1 \equiv a(p_2 - p_1) \pmod{26}$.
    - If $p_2 - p_1$ is a unit modulo 26 Eve may multiply by its inverse to find a, and then solve for b.
    - For example, if Eve knows the word **to** encodes to the string **JS**, then she obtains the equations $9 \equiv 19a + b \pmod{26}$ and $18 \equiv 14a + b \pmod{26}$. Subtracting yields $-9 \equiv 5a \pmod{26}$, whose unique solution is $a \equiv 19 \pmod{26}$; we then also can see that $b \equiv 9 - 19 \cdot 19 \equiv 12 \pmod{26}$.
    - If $p_2 - p_1$ is not a unit then Eve may have to work harder: one can verify that if it is even (and not zero) there will be two possible values of a differing by 13. Only one of these will be odd, so again Eve can find the correct value of a and thus of b. But if $p_2 - p_1$ is equal to 13, then there are many solutions.
  - Attack 3, chosen-plaintext: If Eve has access to the encoding mechanism and can input a plaintext, she can encode the message **ab**, and from the results she can determine the key.
    - As above, the point is Eve can extract the key from two well-chosen points. Specifically, **a** gives the value of $f(0)$ and the encoding of **b** gives the value of $f(1)$.
    - Since $f(0) = b$ and $f(1) = a + b$, Eve can compute $a = f(1) - f(0)$ and $b = f(0)$ directly from this information.
  - Attack 4, chosen-ciphertext: As with the chosen-plaintext attack, Eve can ask for the decoding of **AB**, and as with the chosen-plaintext attack she can use this to determine the decryption function.

- We can further generalize the affine cipher by using an arbitrary function f from the set $\{0, 1, \ldots, 25\}$ to itself, rather than just those functions that are linear modulo 26.
  - In order for this function to be decodable, we need it to be one-to-one (injective).

- An equivalent way to think of this substitution cipher is that it replaces each letter of the plaintext in some predetermined fashion (e.g., **a** $\mapsto$ **W**, **b** $\mapsto$ **C**, and so forth).
  - It is a simple count to see that the number of such functions is $26! \approx 4.03 \cdot 10^{26}$: this is simply the number of ways to permute the 26 possible symbols.
  - Obviously, a brute force decoding will not be effective against a general substitution cipher: even simply to store all of the possible decodings of a 100-character message would take approximately $4 \cdot 10^{16}$ terabytes of storage space. (For comparison, the total amount of internet traffic worldwide in 2013 was estimated to be on the order of $4 \cdot 10^7$ terabytes.)

- If Eve has access to the encoding or decoding mechanism, or to a plaintext/ciphertext pair, she can easily extract the 26 letter substitutions underlying the cipher.
  - For example, if Eve has the encoding mechanism, she can ask it to encode the message **abcdefgh...z** to determine the encoding of each letter.
  - If she has the decoding mechanism, she can ask it to decode **abcdefgh...z**: she can then decode any message.
  - If she has a plaintext/ciphertext pair, she needs only find each letter in the plaintext and corresponding ciphertext to make a table of the encodings.

- Less obvious is how Eve can attack a substitution cipher if she only has the ciphertext: a brute-force method is obviously not going to succeed given the total number of possible keys.

- The standard method for attacking a substitution cipher is a procedure known as frequency analysis: since each appearance of any given letter in the ciphertext always corresponds to the same letter in the plaintext, if we count the frequency of each letter in the ciphertext, we can compare them to the typical letter frequencies in English to try to find the correspondence.
  - Here are the frequencies of the 26 English letters (the nonsense phrase "etaoin shrdlcu" is often used as a mnemonic to remember the order of the common letters):

Letter	e	t	a	o	i	n	s	h	r	d	l	c	u
Frequency	12.7%	9.1%	8.2%	7.5%	7.0%	6.7%	6.3%	6.1%	6.0%	4.3%	4.0%	2.8%	2.8%
Letter	m	w	f	g	y	p	b	v	k	j	x	q	z
Frequency	2.4%	2.4%	2.3%	2.0%	2.0%	1.9%	1.5%	1.0%	0.8%	0.2%	0.2%	0.1%	0.1%

  - Presuming the ciphertext is sufficiently long, counting the frequency of letters allows the original message to be partially decoded.
  - To get further, we can repeat the frequency analysis with pairs of consecutive letters ("digrams") or triples ("trigrams"): we can search for pairs or triples of consecutive letters that occur in the text and compare them to a list of common English digrams and trigrams.
  - In English, the most common digrams are **th** and **he**, followed by **in**, **an**, **er**, **re**, **ed**, **on**, **es**, **ea**, **ti**, **st**, **en**, **at**. For doubled letters, **ll**, **tt**, **ee**, **ss** are the most common. The most common trigram is **the**, followed by **ing**, **and**, **ere**, **her**.
  - Once the ciphertext has been partially decoded, the rest of the message can generally be inferred from context (i.e., from partly decoded words that have only one obvious possible completion). For example, the partly decoded word **Zotato** is probably **potato**.
  - Actually decoding a substitution cipher usually requires some amount of trial and error or guesswork, since short messages will often have letter frequencies that do not actually match the averages for typical English text.

- Example: Decode the message below, which is encrypted with a substitution cipher.

```
KB TI BQ NBK KB TI KRZK PF KRI XAIFKPBN DRIKRIQ KPF NBTHIQ PN KRI YPNC KB FAWWIQ
KRI FHPNOF ZNC ZQQBDF BW BAKQZOIBAF WBQKANI BQ KB KZGI ZQYF ZOZPNFK Z FIZ BW
KQBATHIF ZNC TJ BEEBFPNO INC KRIY
```

  - We first compute the frequencies of the letters in the message. Of the 404 characters, there are 19 occurrences of **K**, 18 of **B**, 17 of **I**, 13 of **F**, 12 of **N**, 11 of **Q** and **Z**, and fewer than 10 of each remaining letter.
  - For digrams, there are 6 occurrences of **KR** and **RI**, 5 of **PN**, and 4 of **KB**, **QK**, and **NC**. There are also 5 occurrences of **KRI**.
  - These facts suggest that **K** is **t**, that **R** is **h**, and that **I** is **e**. Making these substitutions produces the following:

```
tB Te BQ NBt tB Te thZt PF the XAeFtPBN DhetheQ tPF NBTHeQ PN the YPNC tB FAWWeQ
the FHPNOF ZNC ZQQBDF BW BAtQZOeBAF WBQtANe BQ tB tZGe ZQYF ZOZPNFt Z FeZ BW
tQBATHeF ZNC TJ BEEBFPNO eNC theY
```

  - The letter **B** is likely to be the among the letters **a**, **i**, **o**, **n**, **s**. Based on the words it already appears in, such as **tB**, it is most likely to be **o**. Similarly, the letter **Z** is likely to be **a** since it is common and appears in the word **thZt**, and also is a word by itself.

```
to Te oQ Not to Te that PF the XAeFtPoN DhetheQ tPF NoTHeQ PN the YPNC to FAWWeQ
the FHPNOF aNC aQQoDF oW oAtQaOeoAF WoQtANe oQ to taGe aQYF aOaPNFt a Fea oW
tQoATHeF aNC TJ oEEoFPNO eNC theY
```

  - At this stage we can try to complete partial words like **DhetheQ** using a dictionary search (which returns only the one possibility **whether**), which in turn will lead us to other reasonable guesses for words like **aQQoDF**/**arrowF** (**arrows**) and **PF**/**Ps** (**is**):

```
to Te or Not to Te that is the XAestioN whether tis NoTHer iN the YiNC to sAWWer
the sHiNOs aNC arrows oW oAtraOeoAs WortANe or to taGe arYs aOaiNst a sea oW
troATHes aNC TJ oEEosiNO eNC theY
```

  - At this point we essentially have enough letters in most of the words to be able to fill in the remainder, which we can recognize as the beginning of Hamlet's famous soliloquy:

```
to be or not to be that is the question whether tis nobler in the mind to suffer
the slings and arrows of outrageous fortune or to take arms against a sea of
troubles and by opposing end them
```

- We will again remark that although the substitution cipher is somewhat more challenging to break by hand than the affine or shift ciphers, it is still not secure by any means.

1.4 Polyalphabetic Ciphers, Block Ciphers, and Variants

- In this section we will discuss a few other examples of ciphers which are classical attempts at strengthening the substitution and shift ciphers we have discussed.
- Ultimately, although the substitution cipher is resistant to brute-force attacks, the main weakness is that each letter in the plaintext is always encoded the same way in the ciphertext: this provides too much information, because the ciphertext will have similar letter-frequency properties as standard text written in the same language.
- A number of methods, which generally fall under the name polyalphabetic ciphers, have been devised to try to get around this problem: the ultimate idea is to use different alphabets for encoding different letters of the ciphertext, rather than encoding every letter in the same way.
- One way to do this is simply to encrypt multiple letters at a time in a single block, in order to try to avoid being susceptible to the kind of single-letter counting that the substitution cipher is vulnerable to: such ciphers are generally called block ciphers. Other methods instead attempt to flatten out the frequency distribution using different encoding procedures, or change the encoding alphabet in some less predictable way.
- Another class of methods, called transposition ciphers, will rearrange the plaintext characters in a predetermined way so as to jumble up the message.
    - It is of course possible to combine the two procedures of encoding letters with different alphabets and rearranging the letter orders.
- There are many more cryptosystems in existence of the same general flavor as the ones we will discuss. They are all fairly similar, relatively weak from a modern perspective, and primarily interesting only for historical reasons, so we will not try to make an exhaustive study of all of them.
- Like with the substitution cipher, however, these systems generally all suffer from the weakness that if Eve has access to the encoding or decoding mechanism for most of these ciphers, or to a number of plaintext/ciphertext pairs, she can usually extract the key with far less effort than if she only had a ciphertext to work with.
    - Although these may not seem to be such significant problems, in actual historical practice most successful attacks on cryptosystems such as the ones we discuss were achieved using a combination of all of these techniques.
    - Many of these cryptosystems were actually deployed during wars, and there were many cases where weak encryption (or poor security or incorrect use of the system by the soldiers deploying it) was a significant source of intelligence for the opposing forces.
    - The famous Enigma machine, used by the Germans in World War II, was thought to be so complicated as to be unbreakable. However, in a combination of poor operator procedure leading to the leaking of plaintext/ciphertext pairs, capturing parts of the machine to examine the encoding and decoding mechanisms, and a lot of clever reverse-engineering, the Allies were able to break the encryption and read the Axis messages.

1.4.1 The Vigenère Cipher

- One way to try to improve on the basic substitution cipher is the Vigenère (keyword) cipher, which is a modification of the Caesar shift using a vector of length n to encode blocks of n letters at a time.

- This cryptosystem was actually first described in the 1550s in Italy, but it was later attributed to Blaise de Vigenère (a Frenchman who actually invented several other stronger cryptosystems in the late 1500s) in the 19th century, and the name stuck.

- Here is the procedure for the Vigenère cipher:
  - First, we choose a keyword, which (numerically) is a vector of some length n.
  - We then break the message into letter blocks of length n, and then encrypt each block of letters by adding the keyword vector to it. We then put all of the blocks together in the appropriate order.
  - To decrypt, we simply do the inverse: break the ciphertext into blocks of length n and subtract the keyword vector.

- Example: Encode the message **twentysix** using the Vigenère cipher with keyword **one**.
  - Here is a table of the encryption procedure:

Plaintext	t	w	e	n	t	y	s	i	x
#	19	22	4	13	19	24	18	8	23
Key letter	o	n	e	o	n	e	o	n	e
Key #	14	13	4	14	13	4	14	13	4
Encoding	7	9	8	1	6	2	6	21	1
Ciphertext	h	j	i	b	g	c	g	v	b

  - Thus we obtain the ciphertext **HJIBGCGVB**.
  - Note that the letter **t** appears twice in the plaintext, but is represented in the ciphertext by two different characters: the first time by **H** and the second by **G**.
  - Inversely, the letter **G** appears twice in the ciphertext, but represents different letters from the plaintext.

- Example: The encoding of the message

  ```
  to be or not to be that is the question whether tis nobler in the mind to suffer
  the slings and arrows of outrageous fortune or to take arms against a sea of
  troubles and by opposing end them
  ```

  under the Vigenère cipher with keyword **eleven** is

  ```
  xz fz se rzx os oi elvx vw elz uhidxdsa asiolrv emn rbfwim ma xsi hmah es nysjpv
  olr wwmikf eyh vveshw jj byevvkrsfw asexfrz se xz xvor ecqn etetrnx n wpe jj
  gvzywprw lry fl satjwvrr iih glpq
  ```

- Observe that the Vigenère cipher is equivalent to encoding each letter of the plaintext using a different Caesar shift, according to the terms in the keyword.
  - If we take a key of length $n = 1$, then the procedure is the same as the Caesar shift.
  - In the example above with a key length of 3, the 1st, 4th, 7th, 10th, ... , $3n+1$st letters of the plaintext are all encoded with the Caesar shift corresponding to **o**, while the 2nd, 5th, 8th, 11th, ... , $3n+2$nd letters are all encoded with the Caesar shift by **n**, and the 3rd, 6th, 9th, ... , $3n$th letters are all encoded with the Caesar shift by **e**.

- It is rather less obvious what Eve should do if she only has the ciphertext.
  - The Vigenère cipher resisted efforts to break it for centuries, but it can, in fact, be broken.
  - A direct frequency analysis approach is unlikely to succeed: although **e** is still likely to be the most common letter in the plaintext, it can be encoded as several different characters in the ciphertext depending on which letter of the key it is encoded with.
  - If Eve can determine the length of the key, however, then she is essentially reduced to the much simpler problem of decoding several interleaved Caesar shifts.

- For example, if Eve learns that the key has length 3, then she can separate the ciphertext into the 3 pieces each corresponding to one letter of the key, and then do a frequency analysis to determine the most likely possibilities for each of the three Caesar shifts.
- Even a comparatively short text should yield only a small number of possibilities, because of the unevenness of letter distributions (there are not likely to be a large number of **q**s or **z**s in a plaintext of substantial length, for example).

- There are several procedures for determining the length of the key. They each rely on the observation that if the ciphertext is sufficiently long, the same word or partial word is likely to appear encoded in the same way by the key, since there tend to be repetitions of the same letter combinations in written language (the word "the" appears very often, for example).

- The first procedure relies on finding "agreements" between shifts of the ciphertext with the original ciphertext:
  - To test for a key of length k, translate the ciphertext ahead by k characters. Then count the number of places that the translated ciphertext agrees with the original ciphertext. A large number of agreements with a shift of k characters suggests that the key has length k.
  - For example, if the text is **ABAXMQAMZMRT** and $k = 2$ then there are two such agreements (underlined):

A	B	A	X	M	Q	A	M	Z	M	R	T		
		A	B	A	X	M	A	A	M	Z	M	R	T

  - The idea underlying this method is that if we take a sample of text that is Caesar-shifted forward by i letters and a second sample shifted forward by j letters, then the samples will have far more agreements when $i = j$ than otherwise.
  - In our situation, if we translate forward by the same length as the key, then each comparison will compare letters that have been shifted by the same amount, and are therefore more likely to be the same.
  - To explain why this works, write the letter distribution frequencies for English text as a vector of length 26: $\mathbf{v}_0 = \langle a, b, c, \ldots, z \rangle$. The letter distribution for a sample of text that is Caesar-shifted forward by i letters will be given by the vector $\mathbf{v}_i$ obtained by shifting the elements in $\mathbf{v}_0$ forward by i slots.
  - Consider the proportion of times that a Caesar-i shifted sample of English text would have a letter agree with a Caesar-j shifted sample of English text: it is equal to the sum of the number of times each text has the letter a, plus the number of times each text has the letter b, and so forth, which is simply a lengthier description of the dot product $\mathbf{v}_i \cdot \mathbf{v}_j$.
  - Now observe that if we fix i and let j vary, the largest value of this dot product occurs when $j = i$.
  - Intuitively, this is true because the vectors each have the same elements, but permuted: the dot product $\mathbf{v}_i \cdot \mathbf{v}_i$ pairs up all of the largest elements together, whereas $\mathbf{v}_i \cdot \mathbf{v}_j$ will pair large elements of $\mathbf{v}_i$ with smaller ones in $\mathbf{v}_j$, yielding a smaller dot product.
  - More rigorously, we can appeal to the geometric fact that for any two vectors, it is true that $\mathbf{v}_i \cdot \mathbf{v}_j = ||\mathbf{v}_i|| \, ||\mathbf{v}_j|| \cos(\theta)$ where θ is the angle between the vectors and $||\mathbf{v}||$ denotes the length of the vector $\mathbf{v}$. (In general vector spaces, this fact is a corollary of the Cauchy-Schwarz inequality for inner product spaces.) Since each vector has the same length, the dot product is maximized when the vectors are parallel (i.e., equal).

- Example: Find the most likely key length of the following message encoded with a Vigenère cipher:

```
xz fz se rzx os oi elvx vw elz uhidxdsa asiolrv emn rbfwim ma xsi hmah es nysjpv
olr wwmikf eyh vveshw jj byevvkrsfw asexfrz se xz xvor ecqn etetrnx n wpe jj
gvzywprw lry fl satjwvrr iih glpq
```

  - Using the first procedure, we count the number of agreements between the ciphertext and its shift forward by k characters:

Shift	1	2	3	4	5	6	7	8	9	10	11	12
Agreements	9	8	5	5	3	13	5	9	5	8	5	6

  - We see a much larger number of agreements with a shift of 6, so the key length is most likely equal to $\boxed{6}$.

- Indeed, it is equal to 6: this is the same message we encoded earlier.

- Another method for finding the key length, which is often more difficult to perform in practice, relies on finding repeated blocks of letters:
  - Search for blocks of 3 or 4 letters that appear multiple times in the ciphertext. If we assume that these repeated blocks correspond to the same word in the plaintext, then these corresponding blocks are each encoded in the same way by the key, and therefore the difference in their positions is a multiple of the key length.
  - Finding several such duplicated sequences (of sufficient length that it is unlikely they occurred by chance) and then taking the gcd of the shift lengths will give a list of candidates for the key length, and then each of these possibilities can be analyzed separately.

- Once Eve has the key length, she can study each of the individual Caesar shifts separately.
  - This can be done by hand using frequency analysis: the most common two letters are likely to each be one of **etaoin**, and there are very few possible Caesar shifts that can make that happen.
  - Furthermore, the letters **vwxyz** are fairly uncommon but are followed by the common letters **abcde**, so searching for a cluster of uncommon letters followed by a cluster of common letters is also likely to successfully identify the correct shift.
  - Using dot products, we can make the procedure a bit more algorithmic. Specifically, if $\mathbf{w}$ is the vector of letter frequencies that are all encoded with the same letter of the key, then by assumption $\mathbf{w}$ will be close to (a multiple of) one of the vectors $\mathbf{v}_i$. By our analysis above, the dot product will be maximized when $i = j$, meaning that the value of j maximizing the dot product $\mathbf{w} \cdot \mathbf{v}_j$ is the most likely key letter.

- Example: Identify the key used to encode the following message with a Vigenère cipher:

```
aonqsuuoffselttmxlzttquxlsfusqdhqflhytuerrilqdmqahqymqktaeyimedflhzluzkvhnpmvuvweajr
btdmkhvuersuauzqsuaofmohhryeejhizexdzemajwyognphzazpfbvpbawlugqzhwoeyfsgpefawolebzsp
vrqmrgiymephlpfawdfwqqrgahqtidytmolhhnpflhahagwdudzmxxyaxelrjkefldafxqwkpstqmuaofuwd
jozeyptafusqkehaywsyfafhdietigaopuiwvsxqisaoexihwpqdgkhnoqxrkrqmqdfttqvhzttqvxifadmq
ahmfwolebajglaftakhtpdidtsymcfvmqilhuwqteylstgjisepajiahueqrytmxgrplygwwnihqyvwagei
```

  - First we find the key length. We count the number of agreements between the ciphertext and its translate forward by k characters:

Shift	1	2	3	4	5	6	7	8	9	10	11	12
Agreements	13	10	17	18	18	34	19	14	13	9	15	22

  - There is a large peak at 6, so the key probably has length 6.
  - Now we break the ciphertext apart into the 6 Caesar-shifted pieces:

```
aulzldyiakmzhvbvaahhzyzvuoplvilfayhauyjapajtksdavawhkfziallhtvulsaypnw
oottshtlhtelnwtuuorieoapgeeerypwhtnhdakfsooaeyiosopnrttfheatsmwsehtlia
nfttfquqqadupedezfyzmgzbqyfbqmfqqmpazxextfzfhfepxeqoqttambfpyqqtpumyhg
qfmqufedyefzmamrqmeeanpazfazmeaqtofgmefqqueuaatuqxdqmqqdfatdmitgaexgqe
ssxuslrmmylkvjkssoexjpfwhswsrpwrillwxllwmwysyfiiiigxqvvmwjaiclejjqgwyi
uelxqhrqqihvurhuuhjdwhblwgopghdgdhhdxrdkudpqwhgwshkrdhxqogkdfhyiirrwv
```

  - We want to determine the most likely Caesar shift for each piece.
  - For the first piece, the frequency list is $\{12, 1, 0, 2, 0, 2, 0, 6, 3, 2, 3, 7, 1, 1, 1, 3, 0, 0, 2, 2, 4, 6, 2, 0, 5, 5\}$.
  - It might seem that the first letter is probably to be **e**, but this would give very few **i**, **r**, **s**, **t**. If instead the first letter on the list is **t**, then there would be very few **uvwxyz** (as expected) and also a reasonable number of **abcdef** (also as expected). This corresponds to the key letter **h**.
  - For the second piece, the frequency list is $\{5, 0, 0, 1, 8, 2, 1, 6, 3, 0, 1, 3, 1, 3, 8, 3, 0, 3, 5, 10, 2, 0, 3, 0, 2, 0\}$. Testing the various possibilities suggests this corresponds to the key letter **a**, since this puts the largest numbers in the slots of **e**, **o**, and **t**.
  - We can perform the same analysis for the remaining four pieces to see that the key is **hammed**. (The original message is a lengthier portion of Hamlet's soliloquy.)

1.4.2 Running-Key Ciphers and One-Time Pads

- The major weakness of the Vigenère cipher that we exploited was the fact that its key has a finite length: ultimately, once we can compute the key length, we can pull the cipher apart into a separate list of Caesar shifts that we can in principle decode using frequency analysis.
- A way to try to avoid this problem is instead to use a running-key cipher: rather than having a finite key that we repeat over and over to do our encodings, we instead use a key of unbounded length that is never repeated. Traditionally, an easily accessible text is generally used as the key (e.g., a novel, a reference book, or a religious tome).
- For example, if we decide to use the beginning of Anna Karenina as our key ("Happy families are all alike"), the message **thisisatest** would be encoded as follows:

Plaintext	t	h	i	s	i	s	a	t	e	s	t
#	19	7	8	18	8	18	0	19	4	18	19
Key	h	a	p	p	y	f	a	m	i	l	i
Key #	7	0	15	15	24	5	0	12	8	11	8
Encoding	0	7	23	7	6	23	0	5	12	3	1
Ciphertext	A	H	X	H	G	X	A	F	M	D	B

- The running-key cipher is harder to attack than the Vigenère cipher because of the non-repetitive nature of its key. However, with a sufficiently long ciphertext sample, there are still some feasible attacks:
  - One fairly effective procedure is to try to guess parts of the plaintext or the key (which are both hypothesized to consist of normal English words) and subtract them from each possible location in the ciphertext. If the result is intelligible, the probability is high that the resulting pair of words is the correct way to pull apart the plaintext and ciphertext.
  - For example, the words "people", "because", "something", and "through" are all among the 150 most common English words, so if we happen upon the ciphertext **AJSRMOLLNEJ**, subtracting **something** from each of the 3 possible locations produces **ivgnthdyhEJ**, **ArefivedayJ**, and **AJadaksefrd** respectively. It is exceedingly unlikely that a nine-character string would produce another readable portion of a message like "refiveday", so the word **something** is very likely to be in either the plaintext or the ciphertext at that location.
  - Finding enough partially-decoded pieces of the key would then give enough information to allow a computer search for the possible key texts. It is estimated that there are about 130 million different books in the world, and so even an estimate of 30 billion possible keys to test is a computation of size $\approx 2^{35}$, which is computationally feasible with a modern search engine.
  - Another way some information is leaked is through the fact that the running key and the plaintext are both samples of English text. Combining pairs of randomly-chosen English letters will lead to a small imbalance in the letter distributions in the result of the cipher, since for example it is far more likely to draw two random **e**s than two random **j**s.
  - Although the frequency distribution will be much flatter (the 26 one-letter frequencies will range from about 2.6% to 4.8%), it is still ultimately susceptible to some amount of frequency analysis to narrow down the number of possible (key, plaintext) pairs yielding a given portion of the ciphertext.
  - One way to try to resist the first attack would be to apply a Caesar shift, or a substitution cipher, after doing finishing the initial run of the encoding. However, this kind of change can be detected and in principle reversed using frequency analysis.
- The weakness in the running-key cipher lies not in the encryption method itself, but in the choice of the key text: standard text is too predictable.
- If instead we use a randomly-generated sequence of characters, the resulting cryptosystem is called a one-time pad: it is a sequence of random letters used as an encryption key precisely once, and then destroyed.
  - The word "pad" comes from early use of this cryptosystem: the random key sequence was distributed on a pad of paper, thus making it easy to rip off the top sheet and destroy it once it had been used to encode a message.

- In fact, a one-time pad is theoretically unbreakable, provided the sender and recipient are the only ones with a copy of the key.
    - The reason is simple: since the pad's letters are random, the ciphertext is also statistically random, so all possible decodings of a fixed ciphertext into a plaintext are equally likely.
    - For example, the ciphertext **UTUXOJWCZFFYUVT** could have been the plaintext **theanswerisfive** if the key were **bmqxbrayixntmap**, but it could also have been the plaintext **iwantanapplepie** if the key were **mxukvjjckquufnp**.
    - Furthermore, even if part of the plaintext is revealed, the remaining ciphertext still cannot be decoded without the key, because once again all possible results for the undecoded ciphertext are still equally likely.
    - It can be proven, in fact, that any theoretically unbreakable cryptosystem must essentially be equivalent to a one-time pad system.
- However, if a one-time pad is reused, it is no longer secure: the two separate encoded messages can be compared to glean information about the underlying plaintexts. (Hence the name *one-time* pad.)
    - Specifically: given two ciphertexts $\text{cipher}_1 = \text{plain}_1 + \text{key}$ and $\text{cipher}_2 = \text{plain}_2 + \text{key}$, subtracting the two messages will yield $\text{cipher}_2 - \text{cipher}_1 = \text{plain}_2 - \text{plain}_1$: in other words, it completely removes the influence of the random key.
    - In fact, the result is a running-key encryption of one plaintext with the negative of the other, and it is now possible to attack the resulting text using the methods for running-key ciphers we just discussed above.
- One-time pads are occasionally still used in practice, but there are some difficulties.
    - First, the key must be generated randomly. This is actually somewhat difficult: for various reasons, it is a challenging problem to program a computer to produce actually "random" data that does not have any patterns in it.
    - It is also difficult to produce random data by hand, as human-generated random strings statistically look very different from proper random data. For example: try writing down a "random" string of about 100 Ts and Hs. It is very unlikely that there will be any runs of more than 4 identical characters, since (most people feel that) such long runs don't seem "random". On the other hand, flipping an actual fair coin 100 times has about a 97% chance of producing a run of at least 5 identical flips in a row, and about a 55% chance of producing a run of at least 7 identical flips in a row.
    - Furthermore, the keys must be disseminated securely and kept secure for an indefinite period until they are needed for encryption or decryption. This is not a cryptographic problem so much as a practical one, but it is difficult to ensure that the key cannot be retrieved by anyone except the person intending to use it and the person intending to decode the message. It is particularly problematic because the pad must be at least as long as the message itself, and the practical constraints of transporting the information securely are nontrivial.
    - Finally, the keys must be immediately destroyed once they are used so there is no possibility for reuse. Again, this is primarily a practical problem rather than a cryptographic one, but it is difficult to manage given that the keys must also be readily available for use for an indefinite period of time.
    - On the other hand, the system does have the attractive quality of being both provably secure and capable of being implemented fairly practically by hand.

1.4.3 The Playfair and Four-Square Ciphers

- Another simpler cipher that is a modification of the substitution cipher is known as the Playfair cipher.
    - This cryptosystem was invented in the 1850s by Charles Wheatstone and promoted by Lord Playfair (whose name it now bears).

- It was used in the 20th century by the British and Australian militaries due to its simplicity and speed. Amusingly, in the 1850s it was rejected for use by the British government because the officials believed it to be too complicated!

- The Playfair cipher uses a 5×5 letter grid and a 25-letter alphabet, so we will take the common convention of identifying the letters **i** and **j** together. The procedure is as follows:
  - Choose a keyword (or key phrase) and remove all the duplicate letters. Create a 5×5 letter grid, filling in the beginning of the table with the keyword letters and then listing the remaining letters in alphabetical order.
  - Break the plaintext into 2-letter digrams. If any digram is a doubled letter, insert an **x** between the letters and repeat the procedure until there are no duplicates. If the last character is by itself, add an **x** at the end.
  - Encode each digram in the following way: for each pair in the same row, shift each letter to the right (wrapping around the table as necessary). For each pair in the same column, we shift each letter downward (wrapping around as necessary). For all other pairs, replace each letter with the letter in its row and in the other letter's column.
  - To decode, we simply do the reverse: for each pair in the same row we shift each letter to the left, and for each pair in the same column we shift each letter upward. For all other pairs, we replace each letter with the letter in its row and in the other letter's column.

- Example: Encode the message **sleekbaywindow** using the Playfair cipher with keyword **subdermatoglyphic**, and decode the message **HZDOGODZDOAZ**.
  - Here is the encoding grid:

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

  - We group the plaintext into digrams: **sl ee kb ay wi nd ow**. Since there is a doubled letter we place an **x** between the letters (and then must add another **x** at the end), so our list to encode is **sl ex ek ba yw in do wx**.
  - We obtain the pairs **UG DZ DN AY FB CI ET AZ**, so the ciphertext is **UGDZDNAYFBCIETAZ**.
  - To decode, we do the same thing: break the ciphertext into digrams and then apply the reverse procedure.
  - The digrams are **HZ DO GO DZ DO AZ** , which translates back to **on et hr ex et wo**, or **onethrexetwo**. Removing the **x** yields **onethreetwo**.

- The Playfair cipher is quite fast and easy to apply by hand, but it has a number of weaknesses:
  - First, aside from the duplicate letter issues, it is vulnerable to a frequency analysis, since it is simply a substitution encoding on digrams. If the ciphertext is sufficiently long then it is easy to search for the most common digrams and compare them to the common English digrams to decode part of the message directly.
  - We can get more mileage out of this frequency analysis by also observing that reversing the letters in a digram will reverse the letters in the ciphertext: some digrams are common but have uncommon reverses (e.g., **th** is common but **ht** is not) while others also have a common reverse (e.g., both **er** and **re** are common), and comparing the frequency of each digram with its reverse will further help to differentiate them.
  - Once some of the digrams have been decoded, we can use the structure of the encoding grid and the keyword: searching for clusters of 5 letters that often appear paired together will yield likely sets of letters that lie in the same row or column.
  - Also, because the last two rows of the grid are usually a list of less-common letters in alphabetical order, we can search for those letters in the ciphertext to try to determine which letters lie in each column. We can further narrow things down with a small search by trying to guess the terms in the the bottom row, which will usually have most of **quvwxyz** in it.

- With a computer, it can also be quite effective to combine these procedures with a brute-force keyword search.
- There are also other kinds of computational methods such as an iterated hill-climbing procedure: starting with a guess of a random letter grid, the computer searches for small changes (swapping two letters, swapping rows or columns, etc.) that will make the frequency distribution of the decoded text closer to the correct distribution of English letters and digrams. It repeatedly searches for such moves that improve the appearance of the message ("hill-climbing") until it hits a maximum, which will (optimistically) be the actual encoding grid. Repeating this computation and running a dictionary comparison on the end results should eventually yield the correct plaintext.

- A variant of the Playfair cipher is called the four-square cipher: rather than using a single grid, it uses four 5×5 grids to perform the encodings. The upper left and lower right grids are plaintext grids, and the lower left and upper right grids are ciphertext.
  - The encoding procedure is as follows: first, each 5×5 ciphertext grid is filled in according to a keyword, as in the Playfair cipher. The plaintext grids are filled in alphabetical order.
  - Break the plaintext into 2-letter digrams (adding an **x** at the end if necessary), and then encode each digram in the following way:
    - Locate the first plaintext in the upper plaintext grid and the second plaintext letter in the lower plaintext grid.
    - Ciphertext letter 1 is the letter in the upper-right ciphertext grid that is in the same row as plaintext letter 1 and the same column as plaintext letter 2.
    - Ciphertext letter 2 is the letter in the lower-left ciphertext grid that is in the same column as plaintext letter 1 and the same row as plaintext letter 2.
  - To decode, we simply do the reverse: we break the ciphertext into digrams and identify each pair of letters in the ciphertext grids: then plaintext letter 1 is in the same row as ciphertext letter 1 and column as ciphertext letter 2, and plaintext letter 2 is in the same column as ciphertext letter 1 and row as ciphertext letter 2.

- Example: For the four-square cipher with keywords **hydropneumatics** and **pseudomythical**, encode the message **twelvepi** and decode the message **NGDBCKHD**.
  - Here is the encoding grid:

a	b	c	d	e	H	Y	D	R	O
f	g	h	ij	k	P	N	E	U	M
l	m	n	o	p	A	T	IJ	C	S
q	r	s	t	u	B	F	G	K	L
v	w	x	y	z	Q	V	W	X	Z
P	S	E	U	D	a	b	c	d	e
O	M	Y	T	H	f	g	h	ij	k
IJ	C	A	L	B	l	m	n	o	p
F	G	K	N	Q	q	r	s	t	u
R	V	W	X	Z	v	w	x	y	z

  - To encode **twelvepi** we break it into digrams **tw el ve pi**.
  - Now we encode each digram, obtaining **FX HB ZP CH**, so the ciphertext is **FXHBZPCH**.
  - Similarly, to decode **NGDBCKHD**, we break the message into digrams **NG DB CK HD** and then apply the reverse encoding. The result is **gr ee nt ea**, so the plaintext is **greentea**.

- The four-square cipher, although a bit stronger than the Playfair cipher, ultimately suffers from similar weaknesses.
  - There are more possible encodings of digrams, but ultimately it is still it is vulnerable to a frequency analysis, since like the Playfair cipher it is simply a substitution encoding on digrams.
  - Some of the other procedures no longer work (for example, reversing a digram does not reverse the order of the corresponding ciphertext), but ultimately it is still fairly straightforward to break the cipher using a computer.

1.4.4 The Hill Cipher

- Another issue with the Vigenère, substitution, and Playfair ciphers is that changing one letter of plaintext will only change one or two letters of ciphertext.
  - Thus, if a partial plaintext is known or can be guessed, the corresponding ciphertext can be compared to it to extract information about the key.
  - One way to try to avoid such attacks is to encode blocks of plaintext in such a way that changing any one plaintext letter will change many letters in the ciphertext. Such encryption schemes are known as block ciphers.
  - This will reduce the ability to decrypt the message by guessing parts of the plaintext, because there will still be too many decryption possibilities to be computationally feasible.
- An example of a cipher that encodes many letters as a single block is the Hill cipher.
  - This cipher was invented in 1929 by Lester Hill and is based on linear algebra. It was not actually used much in practice, but it is a worthwhile example nonetheless.
- The procedure is as follows:
  - Choose an integer n, the block length, and divide the message into blocks of length n, written as row vectors of length n.
  - Also choose an encoding matrix M, which is an $n \times n$ matrix which is invertible modulo 26.
  - The ciphertext associated to a plaintext vector $\mathbf{p}$ is then given by the matrix product $\mathbf{c} = \mathbf{p}M$.
  - To decode, we simply compute $\mathbf{p} = \mathbf{c}M^{-1}$.
- The only practical difficulty in implementing the Hill cipher is in ensuring the encoding matrix M is invertible modulo 26. We will not go heavily into the details of matrix arithmetic modulo m, but there is a simple criterion for determining when a matrix is invertible:
- Proposition: An $n \times n$ matrix M modulo m has an inverse modulo m if and only if its determinant is a unit modulo m (i.e., is relatively prime to m).
  - Proof: First suppose M has an inverse B. Then because the determinant is multiplicative, $MB \equiv I \pmod{m}$ implies $\det(M)\det(B) \equiv 1 \pmod{m}$, so $\det(M)$ is a unit.
  - Now suppose that $\det(M)$ is a unit modulo m. It is, in particular, not zero, so the inverse of M exists as a real-valued matrix.
  - Furthermore, there is a formula for the inverse matrix: $M^{-1} = \dfrac{1}{\det(M)}\mathrm{adj}(M)$, where $\mathrm{adj}(M)$ denotes the "adjugate" matrix of M, whose (i,j)-entry is the (j,i)-cofactor of M.
  - In particular, the entries in $\mathrm{adj}(M)$ are all integers, and $\det(M)$ is a unit modulo m, so both sides still make sense modulo m: we obtain $M^{-1} \equiv \dfrac{1}{\det(M)}\mathrm{adj}(M) \pmod{m}$, so the inverse exists.
- As a matter of practice, the standard technique of row-reduction (also called Gauss-Jordan elimination) will compute the inverse of a matrix modulo m just as well as it computes the inverse of a matrix with real number entries.
  - For 2×2 matrices we note the explicit formula $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \dfrac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.
- We will note that some implementations of the Hill cipher instead use 29-character alphabet (by adding a space and two punctuation marks such as the period and comma), so as to make it more likely that a random matrix will be invertible.
  - By the proposition, the only way a matrix can be noninvertible modulo 29 is if its determinant is divisible by 29.

- This is far less likely to happen than having the determinant be divisible by 2, which is already enough to make a matrix noninvertible modulo 26.

- Example: Using the Hill cipher with matrix $M = \begin{pmatrix} 5 & 1 & 3 \\ 1 & 4 & 2 \\ 11 & 2 & 7 \end{pmatrix}$, encode the message **attacknow** and decode the message **ORCGYAVAF**.
  - To encode, we write the plaintext as a series of row vectors, yielding $\mathbf{p}_1 = \langle 0, 19, 19 \rangle$, $\mathbf{p}_2 = \langle 0, 2, 10 \rangle$, $\mathbf{p}_3 = \langle 13, 14, 22 \rangle$.
  - Then we right-multiply by the matrix M and reduce modulo 26, yielding $\mathbf{p}_1 M = \langle 20, 10, 15 \rangle$, $\mathbf{p}_2 M = \langle 8, 2, 22 \rangle$, and $\mathbf{p}_3 M = \langle 9, 9, 13 \rangle$.
  - These yield the ciphertext $\boxed{\textbf{UKPICWJJN}}$.
  - To decode, first we compute the inverse matrix M^{-1}: the result is $M^{-1} = \frac{1}{9} \begin{pmatrix} 24 & -1 & -10 \\ 15 & 2 & -7 \\ -42 & 1 & 19 \end{pmatrix}$.
  - Since $9^{-1} \equiv 3 \pmod{26}$ we obtain $M^{-1} \equiv 3 \begin{pmatrix} 24 & -1 & -10 \\ 15 & 2 & -7 \\ -42 & 1 & 19 \end{pmatrix} \equiv \begin{pmatrix} 20 & 23 & 22 \\ 19 & 6 & 5 \\ 4 & 3 & 5 \end{pmatrix} \pmod{26}$.
  - Now we break the ciphertext into row vectors: $\mathbf{c}_1 = \langle 14, 17, 2 \rangle$, $\mathbf{c}_2 = \langle 6, 24, 0 \rangle$, and $\mathbf{c}_3 = \langle 21, 0, 5 \rangle$.
  - We then right-multiply by M^{-1} and reduce modulo 26, yielding $\mathbf{c}_1 M^{-1} = \langle 13, 14, 13 \rangle$, $\mathbf{c}_2 M^{-1} = \langle 4, 22, 18 \rangle$, and $\mathbf{c}_3 M^{-1} = \langle 24, 4, 19 \rangle$.
  - This gives the plaintext $\boxed{\textbf{nonewsyet}}$.

- It is an interesting counting problem to determine the number of invertible $n \times n$ matrices modulo m.
  - Ultimately the answer depends on the prime factorization of m, and is a bit complicated.
  - If for example m is a prime number p, then it can be shown using an argument with vector spaces that the number of invertible $n \times n$ matrices modulo p is $(p^n - 1)(p^n - p) \cdots (p^n - p^{n-1})$.
  - One can then extend this result to show that if m is a prime power p^k then the number of invertible matrices modulo m is $p^{k^2(n-1)}(p^n - 1)(p^n - p) \cdots (p^n - p^{n-1})$.
  - By the determinant criterion, we see that a matrix is invertible modulo 26 if and only if it is invertible modulo 13 and modulo 2.
  - So, using a bit of arithmetic, it can be verified that when $m = 26$, the number of invertible $n \times n$ matrices is $(2^n - 1)(2^n - 2) \cdots (2^n - 2^{n-1}) \cdot (13^n - 1)(13^n - 13) \cdots (13^n - 13^{n-1})$.
  - When $n = 2$ there are 157248 invertible matrices modulo 26, out of a total number of 456976 total: roughly 34%.
  - When $n = 3$ there are about $1.63 \cdot 10^{12}$ invertible matrices modulo 26, out of a total number of about $5.43 \cdot 10^{12}$, roughly 30%.

- Observe that changing one letter of any Hill cipher block will change all the letters of the resulting ciphertext block, provided none of the entries of the matrix M are zero.
  - This provides a measure of security against a direct attack using only a ciphertext. Techniques such as frequency analysis will be relatively ineffective because it is not possible to determine which characters of the ciphertext correspond to equivalent characters in the plaintext.
  - The only situation in which frequency analysis would provide any information is of character blocks the same size as the encoding matrix. If the encoding matrix is 2×2 or 3×3 this will be useful, but not for much larger matrices.
  - If the size of the matrix is relatively small (e.g., smaller than a long English word that is expected to appear multiple times in the plaintext) then by searching for long repeated strings it is possible to obtain possible key lengths and guesses for the decodings of particular common words.

- Ultimately, the Hill cipher is special case of a substitution cipher applied to plaintext blocks, albeit one whose encoding mechanism is vastly easier to store than a general substitution cipher.
- Due to the linearity of the encoding mechanism, however, the Hill cipher is very vulnerable to a known-plaintext attack.
    - In the same way that the affine cipher can usually be decoded by solving a system of equations given two (plaintext, ciphertext) letter pairs, an $n \times n$ Hill cipher can usually be decoded if we are given n^2 (plaintext, ciphertext) letter pairs.
    - Explicitly, each (plaintext, ciphertext) letter pair will yield a linear equation in the n^2 entries of the encoding matrix which may then be solved using linear algebra. If the system is indeterminate (i.e., if some of the letter pair information turns out to be redundant) then adding a few additional letter pairs should yield a unique solution fairly quickly.
- Example: Given that the message **primer** encodes to **MLWILM** under a 2×2 Hill cipher, find the encoding matrix.
    - Breaking the messages into two 2-letter blocks and converting them to row vectors gives us the two equations $\langle 15, 17\rangle M \equiv \langle 12, 11\rangle$, $\langle 8, 12\rangle M \equiv \langle 22, 8\rangle$, $\langle 4, 17\rangle M \equiv \langle 11, 12\rangle$, where all congruences are modulo 26.
    - If $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ then we obtain the six equations $15a + 17c \equiv 12$, $15b + 17d \equiv 11$, $8a + 12c \equiv 22$, and $8b + 12d \equiv 8$, $4a + 17c \equiv 11$, $4b + 17d \equiv 12$.
    - We can write the first four equations in matrix form as $\begin{pmatrix} 15 & 17 \\ 8 & 12 \end{pmatrix} M \equiv \begin{pmatrix} 12 & 11 \\ 22 & 8 \end{pmatrix} \pmod{26}$. However, the matrix $\begin{pmatrix} 15 & 17 \\ 8 & 12 \end{pmatrix}$ on the left is not invertible modulo 26, because its determinant is even.
    - So let us instead take the first two and last two equations: in matrix form we obtain $\begin{pmatrix} 15 & 17 \\ 4 & 17 \end{pmatrix} M \equiv \begin{pmatrix} 12 & 11 \\ 11 & 12 \end{pmatrix} \pmod{26}$.
    - Now the matrix $\begin{pmatrix} 15 & 17 \\ 4 & 17 \end{pmatrix}$ on the left is invertible, since its determinant is $187 \equiv 5 \pmod{26}$, and a bit of arithmetic modulo 26 will eventually produce its inverse as $\begin{pmatrix} 19 & 7 \\ 20 & 3 \end{pmatrix}$.
    - Left-multiplying by this matrix yields $M \equiv \begin{pmatrix} 19 & 7 \\ 20 & 3 \end{pmatrix} \begin{pmatrix} 12 & 11 \\ 11 & 12 \end{pmatrix} \equiv \boxed{\begin{pmatrix} 19 & 7 \\ 13 & 22 \end{pmatrix}} \pmod{26}$.

1.4.5 The ADFGX and ADFGVX Ciphers

- All of the ciphers we have discussed so far have been polyalphabetic ciphers, which simply encode each letter or block of letters in the plaintext according to some pre-chosen collection of alphabets.
- One way to try to avoid the frequency-analysis issues of the other ciphers we have mentioned is to rearrange the letters in the ciphertext in some predetermined fashion, so as to make a frequency analysis more difficult to work out properly.
- One cipher that involves rearranging letters of a ciphertext is known as the ADFGX cipher.
    - This cipher combines a letter substitution (using the letters ADFGX, which were chosen because they were hard to mix up in Morse code) with a rearrangement.
    - The ADFGX cipher was used by the Germans in the early 20th century, including during World War I. At the time, they believed it to be unbreakable, although of course it is not: in fact, it was broken by a French army lieutenant working in the Bureau du Chiffre (literally, "cipher bureau") in mid-1918, well before the end of the war.

- The procedure for the ADFGX cipher involves a 5×5 letter grid (with the same convention as the Playfair cipher, with **i** and **j** identified) as well as a keyword, as follows:
  - Choose a 5×5 letter grid and label the rows and columns with the letters **ADFGX**.
  - Encode each letter of the plaintext using the digram formed by the label of its row followed by the label of its column. This will yield a string of the letters ADFGX.
  - Next, write the letters of this string in a new grid whose columns are labeled by the letters of the keyword (removing duplicates), writing the consecutive letters in rows.
  - Now rearrange the columns so as to put the keyword letters in alphabetical order.
  - Finally, read the message downward in columns.
  - To decrypt a message, simply reverse the procedure: determine the number of letters in each column using the keyword letters and the length of the ciphertext (some columns will have 1 letter more than others). Place the letters into the columns appropriately, un-permute them, and then convert the resulting string of ADFGX letters back into plaintext using the matrix.

- Example: Encode the message **mediumkiwis** and decode the message **FAXFFFGXAFFXDF** using the ADFGX cipher with keyword **square** and the matrix below:

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

  - We first write down the string corresponding to the plaintext letters by looking them up in the grid. This yields **DXFFGAAAAFDXGGAAFDAAFG**.
  - Now we write the letters in the grid associated to the keyword, and then put the column labels in alphabetical order:

s	q	u	a	r	e
D	X	F	F	G	A
A	A	A	F	D	X
G	G	A	A	F	D
A	A	F	G		

a	e	q	r	s	u
F	A	X	G	D	F
F	X	A	D	A	A
A	D	G	F	G	A
G		A		A	F

  - Finally, we read off the message down the columns: **FFAGAXDXAGAGDXDAGAFAAF**.
  - To decode the message **FAXFFFGXAFFXDF**, we count a total of 14 characters: thus, 2 columns have 3 characters (the **s** and **q** columns) and the other 4 will have 2 characters.
  - We can then unscramble the columns to obtain the following:

a	e	q	r	s	u
F	X	F	X	X	D
A	F	F	A	X	F
		G		F	

s	q	u	a	r	e
X	F	D	F	X	X
X	F	F	A	A	F
F	G				

  - Thus, the message string is **XFDFXXXFFAAFFG**, which decodes to **octopus**.

- The ADFGX cipher is more difficult to break than the others we have discussed, primarily due to the rearrangement and the fact that the final message is read off in columns rather than rows. But there are a number of techniques that were ultimately used to break the cipher.
  - If several messages are encrypted with the same key and these messages all begin identically (e.g., "Daily report sent to central command from the installation at ...."), then the two ciphertext matrices, even after rearranging the columns, will have the same first 4 rows or so.
  - Because the ciphertext is then read downward in columns, there will appear blocks of 4 or so letters that appear with almost equal spacing through each of the messages, including at the very beginning of the message. Searching the messages for these common blocks at equal intervals will then determine the keylength (roughly equal to the message length divided by the interval between the common blocks).

- Once the keylength k has been determined, the next task is to unscramble the columns. For any given message we first determine the lengths of all of the columns, since this depends only on the message length and keylength.
    - If we again have a collection of several messages encrypted with the same key that all start identically, then by looking precisely at where each of the "common blocks" starts, we can determine which columns have which lengths for that particular message. By repeating this procedure for messages congruent to 1, 2, ... , $k-1$ modulo k, we can determine the exact column permutation arising from the keyword.
    - For example, if the key length is 6 and one message has length 19, then one column will have length 4 and the others will have length 3. The column of length 4 will correspond to the first letter of the keyword. If another message has length 20, then the two columns of length 4 correspond to the first two letters of the keyword, which allows us to identify the second letter of the keyword. (And so forth.)
- Another method for trying to determine the permutation order of the columns relies on the structure of the substitution grid.
    - If the keyword has an even length, then, because each plaintext character creates two ADFGX characters and there are an even number of columns, the characters in odd-numbered columns will all be associated to row labels and the characters in even-numbered columns will be associated to column labels.
    - It should then be possible to determine which columns are even-numbered and which are odd-numbered by frequency analysis. For example, if the encoding matrix has row **A** given by **bdefs**, while column **A** contains **bmzvw**, then **A** will show up much more often in the odd-numbered columns than in the even-numbered ones. In general, there should be two noticeably distinct letter distributions that each show up in half of the chunks of the message.
    - If the keyword has an odd length, then each column will have row and column labels arranged in an alternating pattern (either rcrcrc... or crcrcr... depending on the lengths of the previous columns in the matrix). Again by doing a frequency analysis, one may determine which ones are which and thus obtain additional information about the correct column positions.
- Once the columns have been unscrambled, the remaining task is to determine the encoding matrix. But this is now easy: each digram from ADFGX corresponds to one letter from the alphabet in a one-to-one fashion, so we are precisely looking to solve a substitution cipher.
- One drawback of the ADFGX cipher is the requirement of a 25-character alphabet. To remedy this, the ADFGVX cipher was created: it is exactly the same as the ADFGX cipher, except the encoding matrix is now 6×6, which allows exactly for 26 letters along with the 10 digits 0-9.
    - Here is an example of an encoding matrix for the ADFGVX cipher:

	A	D	F	G	V	X
A	1	b	c	p	6	v
D	g	t	0	3	q	z
F	j	o	n	8	i	5
G	w	2	d	h	y	u
V	a	s	k	m	r	9
X	l	f	7	e	4	x

- We will close by noting that the cryptanalysis of the ADFGVX cipher is essentially identical to that of the ADFGX cipher (though of course it is slightly harder due to the extra characters).

Well, you're at the end of my handout. Hope it was helpful.